前后端分离的思考与实践（四）

2014/06/20 · Web前端, 开发 · Node.js, XSS, 安全

原文出处： 淘宝UED - lorrylockie

前后端分离模式下的安全解决方案

前言

在前后端分离的开发模式中，从开发的角色和职能上来讲，一个最明显的变化就是：以往传统中，只负责浏览器环境中开发的前端同学，需要涉猎到服务端层面，编写服务端代码。而摆在面前的一个基础性问题就是**如何保障Web安全?**

本文就在前后端分离模式的架构下，针对前端在Web开发中，所遇到的安全问题以及应对措施和注意事项，并提出解决方案。

跨站脚本攻击(XSS)的防御

问题及解决思路

跨站脚本攻击（XSS，Cross-site scripting）是最常见和基本的攻击Web网站的方法。攻击者可以在网页上发布包含攻击性代码的数据，当浏览者看到此网页时，特定的脚本就会以浏览者用户的身份和权限来执行。通过XSS可以比较容易地修改用户数据、窃取用户信息以及造成其它类型的攻击，例如：CSRF攻击。

预防XSS攻击的基本方法是：确保任何被输出到HTML页面中的数据以HTML的方式进行转义（HTML escape）。例如下面的模板代码：

```XHTML
<textarea name="description">$description</textarea>
```

这段代码中的`$description`为模板的变量（不同模板中定义的变量语法不同，这里只是示意一下），由用户提交的数据，那么攻击者可以输入一段包含”JavaScript”的代码，使得上述模板语句的结果变成如下的结果：

```XHTML
<textarea name="description">
</textarea><script>alert('hello')'</script>
</textarea>
```

上述代码，在浏览器中渲染，将会执行JavaScript代码并在屏幕上alert hello。当然这个代码是无害的，但攻击者完全可以创建一个JavaScript来修改用户资料或者窃取cookie数据。

解决方法很简单，就是将`$description`的值进行html escape，转义后的输出代码如下

```XHTML
<textarea name="description">
&lt;/textarea&gt;&lt;script&gt;alert(&quot;hello!&quot;)&lt;/script&gt;
</textarea>
```

以上经过转义后的HTML代码是没有任何危害的。

Midway的解决方案

转义页面中所有用户输出的数据

对数据进行转义有以下几种情况和方法：

1. 使用模板内部提供的机制进行转义

中途岛内部使用KISSY xtemplate作为模板语言。

在xtemplate实现中，语法上使用两个中括号（`{{val}}`）解析模板数据，，默认既是对数据进行HTML转义的，所以开发者可以这样写模板：

```XHTML
<textarea name="description">{{description}}</textarea>
```

在xtemplate中，如果不希望输出的数据被转义，需要使用三个中括号（`{{{val}}}`）。

2. 在Midway中明确的调用转义函数

开发者可以在Node.js程序或者模板中，直接调用Midway提供的HTML转义方法，显示的对数据进行转义，如下：

方法1：在Node.js程序中对数据进行HTML转义

```XHTML
var Security= require('midway-security');
//data from server, eg {html:'</textarea>', other:""}
data.html =Security.escapeHtml(data.html);
xtpl = xtpl.render(data);
```

方法2：在模板中对HTML数据进行HTML转义

```XHTML
<textarea name="description">Security.escapeHtml({{{description}}})</textarea>
```

注意：只有当模板内部没有对数据进行转义的时候才使用Security.escapeHtml进行转义。否则，模板内部和程序会两次转义叠加，导致不符合预期的输出。

推荐：如果使用xtemplate，建议直接使用模板内置的{{}}进行转义； 如果使用其他模板，建议使用Security.escapeHtml进行转义。

过滤页面中用户输出的富文本

你可能会想到：“其实我就是想输出富文本，比如一些留言板、论坛给用户提供一些简单的字体大小、颜色、背景等功能，那么我该如何处理这样的富文本来防止XSS呢？”

1. 使用Midway中Security提供的richText函数

Midway中提供了richText方法，专门用来过滤富文本，防止XSS、钓鱼、cookie窃取等漏洞。

有一个留言板，模板代码可能如下：

```XHTML
<div class="message-board">
{{{message}}}
</div>
```

因为message是用户的输入数据，其留言板的内容，包含了富文本信息，所以这里在xtemplate中，使用了三个大括号，默认不进行HTML转义；那么用户输入的数据假如如下：

```XHTML
<script src="http://eval.com/eval.js"></script><span style="color:red;font-size:20px;position:fixed;">我在留言中</span>
```

上述的富文本数据如果直接输出到页面中，必然会导致eval.com站点的js注入到当前页面中，造成了XSS攻击。为了防止这个漏洞，我们只要在模板或者程序中，调用Security.richText方法，处理用户输入的富文本。

调用方法与escapeHtml类似，有如下两种方式

方法1：直接在Node.js程序中调用

```XHTML
message =Security.richText(message);
var html = xtpl.render(message)
```

方法2： 在模板中调用

```XHTML
<div class="message-board">
Security.richText({{{message}}})
</div>
```

通过调用Security的richText方法后，最终的输出如下：

```XHTML
<div class="message-board">
<span style="color:red;font-size:20px;">我在留言中</span>
</div>
```

可以看出，首先：会造成XSS攻击的`script`标签被直接过滤掉；同时style标签中CSS属性`position:fixed;`样式也被过滤了。最终输出了**无害**的HTML富文本

了解其他可能导致XSS攻击的途径

除了在页面的模板中可能存在XSS攻击之外，在Web应用中还有其他几个途径也可能会有风险。

1. 出错页面的漏洞

一个页面如果找不到，系统可能会报一个404 Not Found的错误，例如：http://localhost/page/not/found

```XHTML
404 NotFound
Page /page/not/found does not exsit
```

很显然：攻击者可以利用这个页面，构造一个类似这样的连接，`http://localhost/%3Cscript%3Ealert%28%27hello%27%29%3C%2Fscript%3E`，并引诱受害者点击； 假如出错页面未对输出变量进行转义的话，那么连接中隐藏的 `<script>alert('hello')</script>` 将会被执行。

在express中，发送一个404页面的方法如下

```XHTML
res.send(404, 'Sorry, we don\'t find that!')
```

这里就需要开发者注意错误页面(404或者其他错误状态)的处理方式。如果错误信息的返回内容带有路径信息（其实更准确的讲，是用户输入信息），就一定要进行escapeHtml了。

后续，错误处理的安全机制，会在Midway框架层面中完成。

Midway解决方案的补充说明

其他模板引擎

Midway默认支持xtemplate模板，但将来也有可能支持其他模板：如jade、mustache、ejs等。目前在主流模板中，都提供了默认转义和不转义的输出变量写法，需要开发者特别留意其安全性。

关于escape的其他支持

除了对页面中输出的普通数据和富文本数据，一些场景中也还包含其他可能需要转义的情况，Midway提供了如下几个常用的转义方法，供开发者使用：

- escapeHtml 过滤指定的HTML中的字符，防XSS漏洞
- jsEncode 对输入的String进行JavaScript 转义 对中文进行unicode转义，单引号，双引号转义
- escapeJson 不破坏JSON结构的escape函数，只对json结构中name和vaule做escapeHtml处理
- escapeJsonForJsVar 可以理解就是jsEncode+escapeJson

例子如下

```XHTML
var jsonText ="{\"<script>\":\"<script>\"}";
console.log(SecurityUtil.escapeJson(jsonText));// {"&lt;script&gt;":"&lt;script&gt;"}
var jsonText ="{\"你好\":\"<script>\"}";
console.log(SecurityUtil.escapeJsonForJsVar(jsonText));//{\"\u4f60\u597d\":\"&lt;script&gt;\"}
var str ="alert(\"你好\")";
console.log(SecurityUtil.jsEncode(str));// alert(\"\u4f60\u597d\")
```

跨站请求伪造攻击(CSRF)的预防

问题及解决思路

名词解释： 表单：泛指浏览器端用于客户端提交数据的形式；包括a标签、ajax提交数据、form表单提交数据等，而非对等于HTML中的form标签。

跨站请求伪造（CSRF，Cross-site request forgery）是另一种常见的攻击。攻击者通过各种方法伪造一个请求，模仿用户提交表单的行为，从而达到修改用户的数据或执行特定任务的目的。

为了假冒用户的身份，CSRF攻击常常和XSS攻击配合起来做，但也可以通过其它手段：例如诱使用户点击一个包含攻击的链接。

解决CSRF攻击的思路分如下两个步骤

解决CSRF攻击的思路分如下两个步骤

1. 增加攻击的难度。GET请求是很容易创建的，用户点击一个链接就可以发起GET类型的请求，而POST请求相对比较难，攻击者往往需要借助JavaScript才能实现；因此，确保form表单或者服务端接口只接受POST类型的提交请求，可以增加系统的安全性。
2. 对请求进行认证，确保该请求确实是用户本人填写表单或者发起请求并提交的，而不是第三者伪造的。

一个正常用户修改网站信息的过程如下

- 用户请求修改信息(1) -> 网站显示用户修改信息的表单(2) -> 用户修改信息并提交(3) -> 网站接受用户修改的数据并保存(4)

而一个CSRF攻击则不会走这条路线，而是直接伪造第2步用户提交信息

- 直接跳到第2步(1) -> 伪造要修改的信息并提交(2) -> 网站接受攻击者修改参数数据并保存(3)

只要能够区分这两种情况，就能够预防CSRF攻击。那么如何区分呢？ 就是对第2步所提交的信息进行验证，确保数据源自第一步的表单。具体的验证过程如下：

- 用户请求修改信息(1) -> 网站显示用于修改信息的空白表单，表单中包含特殊的token同时把token保存在session中(2) -> 用户修改信息并提交，同时发回token信息到服务端(3) -> 网站比对用户发回的token和session中的token，应该一致，则接受用户修改的数据，并保存

这样，如果攻击者伪造要修改的信息并提交，是没办法直接访问到session的，所以也没办法拿到实际的token值；请求发送到服务端，服务端进行token校验的时候，发现不一致，则直接拒绝此次请求。

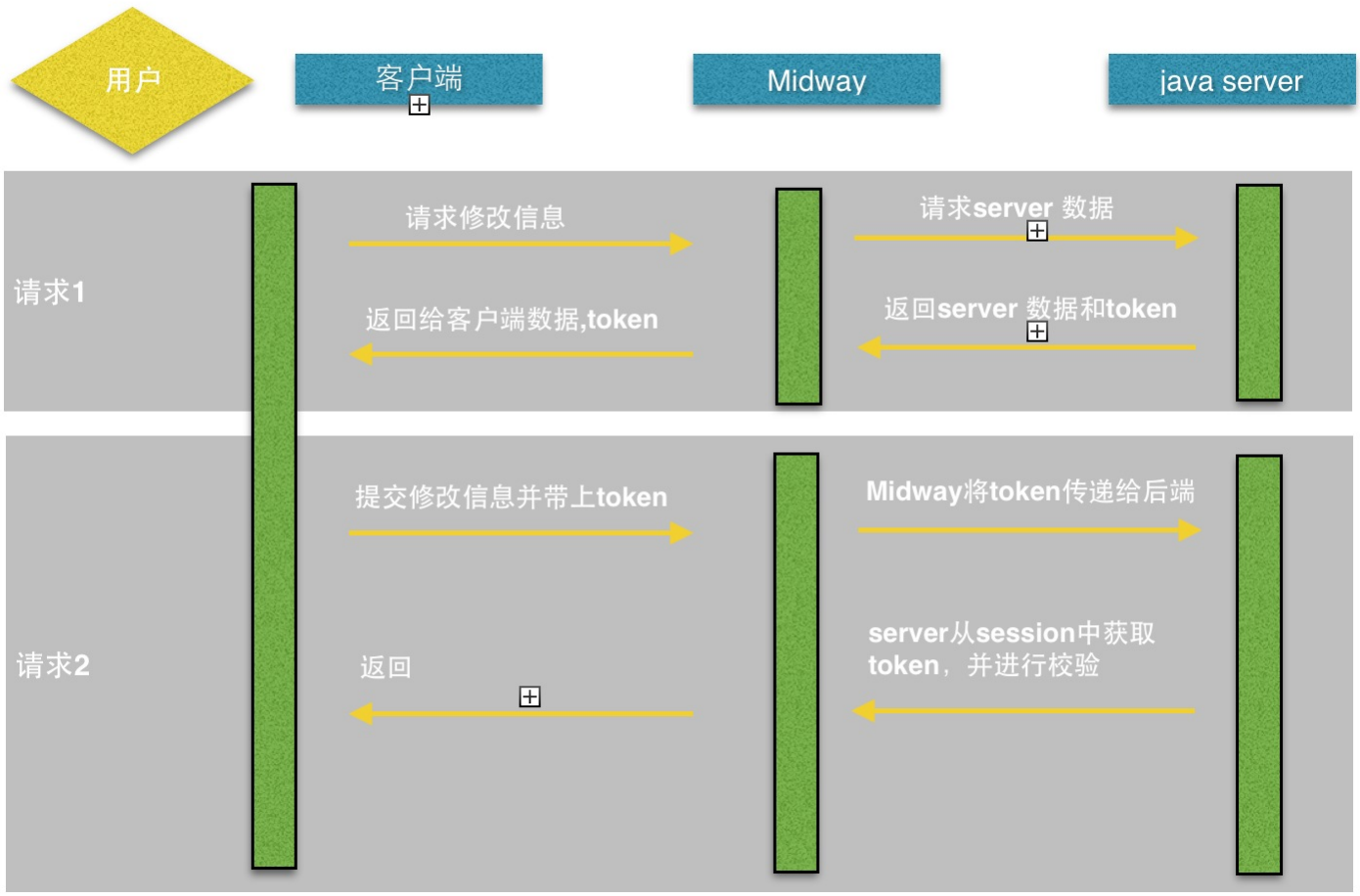
Midway解决方案

禁用GET提交表单

如果服务端不接受GET方式提交的表单数据，那么将会给攻击者带来非常大的难度；因为在页面上构造一个a标签href属性或者img标签src属性来构造一个请求是非常容易的，但是如果要POST提交，就必须要通过脚本才可以实现。

用CSRF token验证请求

因为Midway不涉及到淘宝分布式session及token校验这一层面逻辑，所以在Midway框架中，只将token在server和客户端之间进行转发，本身不做实际的校验工作。流程如下：



后续：在Midway中，Node.js和淘宝的分布式session对接后，可以考虑在Midway这一层自动进行token校验；毕竟安全校验越早进行，成本也会更低。

建议：在Midway中，可以判断是否request中有token的值，如果一个修改操作，没有token，可以直接在Midway层认为是不安全的，将请求丢弃掉。

其他安全问题

关于常见的Web安全问题，还有如下几种，这里只做一些简介，后续会持续继承到Midway framework中。

- HTTP Headers安全
  - CRLF Injection 攻击者想办法在响应头中注入两个CRLF特殊字符，导致响应数据格式异常，从而注入script等
  - **拒绝访问攻击** 每个请求因为都会默认带上cookie，而服务器一般都会限制cookie的大小，这就导致了，如果用户客户端cookie被设置成了超过某个阀值，那么用户就再也无法访问网站了
  - cookie**防窃取** 一般cookie窃取都是通过JavaScript(XSS漏洞)获取到的，所以尽量将cookie设置成http only，并且加上cookie过期时间

关于cookie的安全问题，之前WebX已经有较好的解决方案；此次Midway不负责cookie的设置和校验等工作，只负责转发到WebX层面进行check

关于Node.js

XSS等注入性漏洞是所有漏洞中最容易被忽略，占互联网总攻击的70%以上；开发者编写Node.js代码时，要时刻提醒自己，永远不要相信用户的输入。

比如如下几个例子。

- `var mod = fs.readFileSync('path');` 如果path来源于用户输入，那么假设用户输入`/etc/password`，则会读取到不应该读取的内容，造成密码泄漏风险
- `var result = eval(jsonVal);` 一定要确保jsonVal是json，而不是用户的输入
- …… 其他可能包含用户输入的地方，一定要确认用户的输入是我们期望的值

总结

前后端分离模式下，可以让传统的前端开发人员开始编写后端代码，虽然从架构上讲，只负责模板这一层，但也会接触大量的后端代码；所以安全对于前端来说，这是一个不小的挑战。

【附】相关文章列表

1. 《前后端分离的思考与实践（一）》
2. 《前后端分离的思考与实践（二）》
3. 《前后端分离的思考与实践（三）》

相关文章

- XSS 前端防火墙（4）：天衣无缝的防护
- XSS 前端防火墙（5）：整装待发
- 几种极其隐蔽的XSS注入的防护
- XSS 前端防火墙（1）：内联事件拦截
- XSS 前端防火墙（3）：无懈可击的钩子
- XSS 前端防火墙（2）：可疑模块拦截

可能感兴趣的话题

- 中文出身的妹纸，零基础学习JAVA靠谱么。。。 · 123
- 毕业就失业，Finally It comes to me · 23
- 女生搞开发能干一辈子吗？有没有发展前景啊？ · 50
- 对于互联网IT公司拖欠延期发工资怎么看待？ · 15
- 压力好大，有时候真的想回到老家找个普普通通的工作，不再城市里漂泊了 · 22
- 后端转前端，转型期真心痛苦 · 14

redis3.2新功能 – GEO地理位置命令介绍
2 天前 · 4

JS 又是第一编程语言：GitHub 2016 年度报告
3 天前 · 34 · 3

精选工具资源 更多资源 »

android-remote-notifications：从远程JSON文件拉取...
Android, 通知

PHP CPP：一个开发PHP扩展的C++库
PHP, 扩展

Apache Mahout：经典机器学习算法库
Java, 机器学习

WilliamChart：优美和直观的图表库
Android, 图表

Singularity：方便部署和操作的Mesos框架
Java, 集群管理

最新评论

- Re: 算法系列（2）：三只水桶等分水...
  第一种方法，最后一步：‘从容积是5升的桶中倒3升水到容积是8升的桶中’，是不是错了？
- Re: Linux 中断处理原理分析
  # interrupte# interruptsinterrupttasklet
- Re: 我见过最有趣的代码注释，都在这...
  12不是很信
- Re: PM 叫你去改一个 Bug，后来...
  现在初三，因为这个原因，理想就是不受雇于人，有一家自己的（小）公司。这种活法太恶心
- Re: 你应该知道的计算机网络知识
  同意，因为一般一台机子只配有一个网络适配器(网卡)，所以也就默认一个MAC地址对应一台唯一的电脑...
- Re: 133 行代码实现质感地形
  应该感谢博主搜集了这么好的资源！！辛苦啦--
- Re: 133 行代码实现质感地形
  谢谢提醒。已经在正文更新了 :)
- Re: PM 叫你去改一个 Bug，后来...
  很赞的文章，翻译得也非常赞。解决客户的问题，这个是最大的价值，当然也不必要放弃原则：P

关于伯乐在线博客

在这个信息爆炸的时代，人们已然被大量、快速并且简短的信息所包围。然而，我们相信：过多“快餐”式的阅读只会令人“虚胖”，缺乏实质的内涵。伯乐在线内容团队正试图以我们微薄的力量，把优秀的原创文章和译文分享给读者，为“快餐”添加一些“营养”元素。

快速链接
网站使用指南 »
问题反馈与求助 »
加入我们 »
网站积分规则 »
网站声望规则 »

关注我们

新浪微博：@伯乐在线官方微博
RSS：订阅地址
推荐微信号

程序员的那些事 UI设计达人 极客范

合作联系
Email：bd@Jobbole.com
QQ：2302462408 （加好友请注明来意）

更多频道

小组 – 好的话题、有启发的回复、值得信赖的圈子
头条 – 分享和发现有价值的内容与观点
相亲 – 为IT单身男女服务的征婚传播平台
资源 – 优秀的工具资源导航
翻译 – 翻译传播优秀的外文文章
文章 – 国内外的精选文章
设计 – UI,网页，交互和用户体验
iOS – 专注iOS技术分享
安卓 – 专注Android技术分享
前端 – JavaScript, HTML5, CSS
Java – 专注Java技术分享
Python – 专注Python技术分享