# 上一节我们学习了CSRF，了解到它非常危险而且我们的网站没有任何防护措施，这一节我们来学习如何防护csrf

# 1.我们先通过流程图来了解防护原理p178

|  |
| --- |
|  |

## 原来就是使用token技术，对比方法

|  |
| --- |
|  |

## 因为cookie的token我自动获得但是请求头的token需要手动设置，而且我们是用ajax提交的，就算攻击者自动怎么设置，他也无法攻击，因为此时发生跨域请求了。浏览器无法发送请求。这就是ajax请求的好处

# 2.代码实操，把webA项目复制一份，改名：webA\_anti\_csrf，注意是使用父文件夹的node依赖的p179

|  |
| --- |
|  |

# 为了更好的对比学习，我们把没有csrf防护的两个网站放到一个叫做no\_anti\_csrf的文件夹中，把有防护措施的网站放在with\_anti\_csrf文件夹中，其实webB是没有变化的，我们只修改webA项目，还是使用父文件夹的node依赖

|  |
| --- |
|  |
|  |

# 为了方便对照，我们把app.js改名01-anti-csrf-app.js

# 具体操作的代码实现如下，需要额外做3步工作：1）设置token到cookie中，2）发送请求时，需要在请求头提交token信息，3）token验证，需要用到cookie-parser，如果没有，需要先安装

## 1.我们需要一个能够生产token的函数，如图

|  |
| --- |
|  |

### 也可以这么写

|  |
| --- |
|  |

## 2.我们先在01-anti-csrf-app.js编写这个函数，

|  |
| --- |
|  |

## 3.然后测试一下，只是刷新页面，并没有点击登录按钮

|  |
| --- |
|  |
|  |

### 可见生成了token

## 4.把测试代码注释了，我们来到/transfer路由的get处理代码里面，在这里添加设置token到cookie的代码，注意需要安装cookie-parser，需要引入中间件并且在app中全局注册这个中间件

|  |
| --- |
|  |

## 5.修改views/transfer.html在ajax请求的请求头里面添加token设置，需要先定义一个getCookie函数

|  |
| --- |
|  |

### 然后我们在views/transfer.html在ajax请求的请求头里面修改代码

|  |
| --- |
|  |

## 6.在/transfer路由的post方法处理代码里面添加token获取和验证他们是否相等

|  |
| --- |
|  |

### 注意：获取请求头中的token有2种方法，1）使用req.get方法，需要跟你定义时候的书写完全一样，另外一种是使用cookie-parser，直接通过req.headers[键名],不过注意，cookie-parser会把所有请求头的key全部变为小写，需要使用req.headers['x-csrftoken']来获取X-CSRFToken

## 7，为了更好的用户体验，我们把transfer.html文件修改一下，当ajax post提交成功响应，清空文本框的内容并且跳转到一个叫做/done的路由

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>转账页面</title>  </head>  <body>      <h3>转账</h3>      <hr>      <form method="post">          <p>账号：<input type="text" name="account" id="account"></p>          <p>金额：<input type="number" name="money" id="money"></p>          <p>              <input type="button" value="转账" id="btn">          </p>      </form>      <!-- action="/transfer" -->      <script src="/js/jquery3.1.0.min.js"></script>      <script>          $('#btn').click(function (e) {              let account = $('#account').val()              let money = $('#money').val()              // alert(account)              let params = {                  account,                  money              }              //发送ajax post请求              $.ajax({                  url: '/transfer',                  method: 'post',                  contentType: 'application/json',                  data: JSON.stringify(params),                  headers: { 'X-CSRFToken': getCookie('csrf\_token') },                  success: function (data) {                      console.log("data", data);                      alert(`成功给${account}转账${money}元`)                      //清空输入框的值                      $('#account').val("  ") //有参数是设置值，没有参数是获取值                      $('#money').val("  ")                      $(location).attr('href', '/done') //jquery跳转页面                  }              })              //获取cookie函数：getCookie              function getCookie(name) {                  let r = document.cookie.match("\\b" + name + "=([^;]\*)\\b")                  return r ? r[1] : undefined              }          })      </script>  </body>  </html> |

## 8.然后我们响应在01-anti-csrf-app.js中添加对/done路由的处理代码其实就是渲染一个叫做done.html的页面，里面有一个链接可以跳转到首页

|  |
| --- |
|  |

## 9.当然得有这个页面，我们在views文件夹里面新建一个done.html,内容如下

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>转账成功</title>  </head>  <body>      <h3>转账成功，点击链接返回首页: <a href="/">首页</a></h3>  </body>  </html> |

## 10测试一下，先按照正常模式在自己的服务器上测试，一切正常

|  |
| --- |
|  |
|  |
|  |
|  |

## 11.然后我们运行webB的01-webB-app.js文件

|  |
| --- |
|  |

## 12.点击按钮，效果如下

|  |
| --- |
|  |

### 操作失败，说明我们发防护措施是有效的.

# 本节学习结束，

## 01-anti-csrf-app.js的完整代码如下

|  |
| --- |
| let express = require('express')  let cookieParser = require('cookie-parser')  let cookieSession = require('cookie-session')  let bodyParser = require('body-parser')  let path = require('path')  let port = 3500  let app = express()  //全局注册bodyParser中间件  app.use(bodyParser.urlencoded({extended:false}))  app.use(bodyParser.json())  //全局注册cookieParser中间件,注意：有的中间件需要小括号，有的中间件不需要  app.use(cookieParser())  //全局注册cookieSession中间件  app.use(cookieSession({      name:'my\_session',      keys:['adefg%%78$$','o9l1##dd09=='] ,//这个其实就是加密用的盐      maxAge:2\*24\*1000\*60\*60 //过期时间为2天  }))  app.use(express.static(\_\_dirname+'/public'))//静态服务器，根据需要配置  //处理中文乱码  app.use(function (req, res, next) {      res.setHeader('Content-Type', 'text/html;charset=utf-8');      next();  });  //配置模板引擎,4步，一个engine和3个set  app.engine('html',require('express-art-template'))  app.set('view options',{      debug:process.env.NODE\_ENV!=='production'  })  app.set('views',path.join(\_\_dirname,'views'))  app.set('view engine','html')  //编写一个能够随机生成token的函数  function getRandomString(n){      let str = ''      while(str.length<n){          str += Math.random().toString(36).substring(2)      }      return str.substring(str.length-n) //注意这里其实就是从0开始截取到最后  }  app.all('/',(req,res)=>{       if(req.method=="GET"){          // console.log(getRandomString(48));          res.render('login')       }else if(req.method=="POST"){           //获取用户名，密码           let {username,pwd} = req.body           console.log(username,pwd);           //模拟数据库数据验证           if(username === 'admin' && pwd === '12345'){                 //验证成功,需要设置状态保持，用来在其他路由验证用户是否登录                 req.session['username'] = username                 //转到transfer路由                 res.redirect('/transfer')           } else{              res.send("用户名或者密码错误！！！")           }         }  })  //转账路由,需要设置csrf\_token  app.all('/transfer',(req,res)=>{       //验证用户是否登录,必须一上来就做，无论用户是get还是post请求都需要登录验证      let username = req.session['username']      if(!username){ //用户没有登录          //重定向到首页执行登录操作          res.redirect('/')      }      //功能进入这里，说明用户已经登录了      if(req.method=="GET"){          //产生csrf\_token并且设置到cookie          res.cookie('csrf\_token',getRandomString(48))          res.render('transfer') //渲染转帐页面      }else if(req.method=="POST"){          //模拟转账          //需要先获取两个token值          let csrf\_token = req.cookies['csrf\_token'] //获取cookie中的token          // let x\_csrf\_token = req.get('X-CSRFToken') //获取请求头中的token          let x\_csrf\_token = req.headers['x-csrftoken'] //获取请求头中的token,cookie-parser会把heder里面的键全部转为小写          // console.log(csrf\_token,x\_csrf\_token);          //然后进行比较验证他们是否是同一个token           if(csrf\_token!==x\_csrf\_token){              //如果不相等，说明是csrf伪造请求，必须终止代码继续往下执行              res.send('csrf\_token验证失败,转账无法进行,去检查你的token')              return           }            //如果代码能够执行到这里说明他们相等，就可以进行转账          let {account,money} = req.body            //模拟转换成功，在后输出信息          console.log(`成功给${account}转账${money}元`);          res.json({account,money})      }    })  app.get('/done',(req,res)=>{      res.render('done')  })  app.listen(port,()=>{       console.log('server is ready :http://localhost:'+port+'/');  }) |

## views/login.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <h3>用户登录</h3>      <hr>      <form method="post" action="/">          <p>账号：<input type="text" name="username" id="username"></p>          <p>密码：<input type="password" name="pwd" id="pwd"></p>          <p>              <input type="submit" value="登录" id="btnLogin">          </p>      </form>  </body>  </html> |

## views/transfer.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>转账页面</title>  </head>  <body>      <h3>转账</h3>      <hr>      <form method="post">          <p>账号：<input type="text" name="account" id="account"></p>          <p>金额：<input type="number" name="money" id="money"></p>          <p>              <input type="button" value="转账" id="btn">          </p>      </form>      <!-- action="/transfer" -->      <script src="/js/jquery3.1.0.min.js"></script>      <script>          $('#btn').click(function (e) {              let account = $('#account').val()              let money = $('#money').val()              // alert(account)              let params = {                  account,                  money              }              //发送ajax post请求              $.ajax({                  url: '/transfer',                  method: 'post',                  contentType: 'application/json',                  data: JSON.stringify(params),                  headers: { 'X-CSRFToken': getCookie('csrf\_token') },                  success: function (data) {                      console.log("data", data);                      alert(`成功给${account}转账${money}元`)                      //清空输入框的值                      $('#account').val("  ") //有参数是设置值，没有参数是获取值                      $('#money').val("  ")                      $(location).attr('href', '/done') //jquery跳转页面                  }              })              //获取cookie函数：getCookie              function getCookie(name) {                  let r = document.cookie.match("\\b" + name + "=([^;]\*)\\b")                  return r ? r[1] : undefined              }          })      </script>  </body>  </html> |

## views/done.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>转账成功</title>  </head>  <body>      <h3>转账成功，点击链接返回首页: <a href="/">首页</a></h3>  </body>  </html> |