XSS Hack：获取浏览器记住的明文密码

余弦（@evilcos）
evilcos@gmail.com
[AT]knownsec team
[AT]xeye team
2012/1

0x01. XSS 获取明文密码的多种方式

我已经感受到 Web 潮流带来的巨大革新，尤其是最近 HTML5 越来越火。浏览器们在客户端瓜分着这个 Web OS，只要是对用户体验好的功能，浏览器之间就会互相学习，然后去实现，但是实现总是存在一些差异，有些差异是用户体验上的，有些则可能带来安全问题。

这篇文章是想深入描述下浏览器记住用户密码这种机制带来的安全问题与实现上的一些差异性。黑客们如何通过技巧获取到浏览器保存的密码，明文。

先回到 XSS 本身上，XSS 获取明文密码的方式有哪些？

1. 通过 DHTML 钓鱼方式

比如 document.write 出一个登陆页面，或一个登录框。也就是目标网站的登录方式是怎样的，就通过 DOM 模拟出怎样的。用户很难区分同域内的钓鱼，如果再次输入密码登录时就悲剧了。

2. 通过 JavaScript hook 住密码框

满足某个事件（如 onsubmit/onblur/onchange 等）就记录之。

3. 通过 JavaScript 实现键盘记录器的功能

监听用户在表单里的击键事件，记录击键的值，有时候这个效果会非常不错。

上面这三个方法都是大家在用的，效果各有千秋，这次我要提一种新的方式：通过利用浏览器保存密码这个机制来达到获取明文密码的目的，效果显得更加直接。

0x02. 浏览器记住密码的机制

现在回到浏览器的这个机制上，最早是哪个浏览器实现的，我懒得去考证了，可能是 IE。在用户登录的时候，浏览器会提示保存密码，可以在下面这个地址在线查看我测试的 8 个浏览器截图：

http://evilcos.me/lab/xss_pwd/

注：这也是本文的 demo 地址，测试的浏览器有：

Chrome(v 16.0.912.75 m)
IE9
IE8
Firefox(v 9)
Opera(v 11.60)
Safari(v 5.1.2)
Maxthon(v 3.2.2.1000)

Sogou(v 3.0.0.3000)

保存密码是为了提高用户体验，省去了每次登录需要输密码的麻烦，在这个机制之前经常是通过身份认证的本地 Cookie 来实现的，也许是因为并不是所有网站都采用持久化 Cookie，浏览器才开始选择了这样的方式。而且现代浏览器大多有一个机制：云同步，除了书签、个人偏好外，还可以同步浏览器记住的密码，使得用户在任意地方都可以同步自己的“习惯”。有的身份认证 Cookie 是绑定 IP 的，这样的话同步 Cookie 就不好使了。简而言之吧，密码这东西就是方便，可也太滥用了，滥用有风险，而且还来了个云同步，黑客兴奋了。

在浏览器记住密码机制之前大家应该都知道还有一个很火热的机制：表单自动填充！曾经出现的安全风险是：由于这个自动填充的值是跨域共享的，攻击者可以在自己的域放一个页面，用户的浏览器访问后，会自动填充这个页面的表单（比如 Email、家庭地址、手机号等等，如果用户的浏览器记住过这些值的话），然后这个页面的 JavaScript 就可以获取到这些值了。这些值还好，攻击者并不一定很喜欢，可是明文密码就不一样了。

记住密码机制需要遵循同源策略，但是如果有 XSS 就可以忽略这个同源策略，注入 JavaScript 去得到这个明文密码:P

下面我以 Chrome 为例深入说明说明，攻击者通过这个机制是如何得到你的明文密码的。Opera 与 IE 的机制相对来说是最安全的，而搜狗浏览器在这方面的安全性最差。

0x03. 获得 Chrome 记住的密码

先来看 Chrome，demo 地址：http://evilcos.me/lab/xss_pwd/。可以输入 admin/1234567，然后 LOGIN 试试。浏览器弹出保存密码提示时，选择保存。重新载入这个 demo 地址，可以看到浏览器已经自动填充了密码。点击按钮“see ur pwd”会弹出你输入的密码明文。

实际上你查看页面源代码是看不到密码的，这个密码是浏览器判断页面加载后，发现表单中有密码项，就自动填充最近一次记录的用户名与密码，就像（或者说就是）一次 DOM 操作，动态填充。既然是 DOM 操作，那么在这之后我们控制 JavaScript 也来一次 DOM 操作，这次是读，将密码项里的 value 值读出来，是不是就得到了明文密码？对……是这样！

知道这个过程后，邪恶的想法诞生了……

这个机制遵循同源策略，那么如果在一个域内，任意页面存在 XSS，就应该可以通过 DOM 动态创建一个包含一模一样的用户名与密码表单项的表单出来，然后等待浏览器自动填充密码后，再通过 DOM 操纵得到密码项里的值。

开始实验！

这个页面 http://evilcos.me/lab/xss_pwd/的表单是这样：

```
<form method="post" action=".">
    <label for="username">USER: </label><input id="username" name="username" type="text" class="text" value="" />
    <label for="password">PASS: </label><input id="password" name="password" type="password" class="text" value="" />
    <input type="submit" class="submit" value="LOGIN" />
    <input type="hidden" name="next" value="" />
</form>
```

浏览器是如何记住这个表单的，以确保唯一性？有几个关键值（不同浏览器有差异，不过影响不大）：

1. 为了遵循同源策略，需要域名：evilcos.me
2. 需要一个<form>标签
3. 需要 id 或 name 为 username 的用户名<input>表单项
4. 需要 id 或 name 为 password 的密码<input>表单项

如果是这样，攻击者发现同域内 XSS 后，就要开始构造一段 payload，这个 payload 用于自动创建出这样的表单，这个表单浏览器要能够认识（认为是之前记住密码的那个表单:P），并且必须在浏览器开始自动填充密码之前出来（否则得不到填充值），最后必须在浏览器填充完密码后开始获取表单项的值（否则获取到的值是空的）。

条件好像很苛刻，哪个步骤时间把握不好，攻击就失败了。针对这个场景我构造了一个 payload，如下：

```
function create_form(user) { /*获取明文密码*/
    var f = document.createElement("form");
    document.getElementsByTagName("body")[0].appendChild(f);
    var e1 = document.createElement("input");
    e1.type = "text";
    e1.name = e1.id = "username";
    e1.value = user;
    f.appendChild(e1);
    var e = document.createElement("input");
    e.name = e.type = e.id = "password";
    f.appendChild(e);
    setTimeout(function () {
        alert("i can see ur pwd: " + document.getElementById("password").value);
    }, 3000); //  时间竞争
}

create_form('');
```

也可以查看 http://evilcos.me/lab/xss_pwd/xssme.html 的代码，create_form 函数的执行优先于整个 document 文档的完全解析，这时会自动创建一个登录表单（和之前记住密码的表单关键部分是一样的，这就足够了），然后等待 3000 毫秒，待整个 document 文档解析结束（此时浏览器已经完成了密码填充），最后获取密码表单项里的值，成功！

3000 毫秒不靠谱就来个 for 循环直到获取到密码值才退出。

0x04. 插一个题外点：时间竞争

这个话题很大，就是谁先谁后的问题，不仅和浏览器解析处理整个 DOM 树的顺序有关系，也和我们要达到的目的有关，比如浏览器解析顺序的一个经典例子：

```
<script src='http://remote/x.js'></script>
<body></body>
```

是先解析完远程的 js 脚本，还是先解析<body>标签？

如果这样呢？

```
<script id='rfi'></script>
<script>
```

```
document.getElementById('rfi').src = 'http://remote/x.js';
</script>
<body></body>
```

我们最好对“时间竞争”心里有数，搞清楚浏览器解析的机制，这样我们的 payload 才能达到我们的目的。

0x05. 各浏览器的差异

我已经习惯差异了，而且喜欢差异，因为这样很可能会带来一些安全问题，不过前端工程师们就不喜欢了:&，下面我只讲关键的差异，那些小的，大家自己试验，自己发现。

1. Safari 浏览器

只有 Safari 默认是关闭这个机制的。如果开启后，效果和 chrome 一样，非常好用！

2. Opera 浏览器

Opera 好像很安全，记住密码后，浏览器并不会自动填充密码，而是要用户自己点击地址栏左边的钥匙图标，才会开始填充并登录。

3. IE8/9 浏览器

IE8/9 及部分这个内核的浏览器（比如遨游的 IE 模式）很聪明，将每个登录表单绑定到所在的页面上（下面简称这个页面为绑定页面），由于绑定页面地址是唯一的，同域内其他页面就无法通过生成一个一模一样的表单来获取密码了。

如果就这样还是不安全:P，因为 XSS 可以动态 iframe 进这个绑定页面，然后注入 JS 进行任意 DOM 操作，同样非常容易获取到密码表单项的值，IE 估计是考虑到了这个，通过 iframe 调用绑定页面也无效。而且 IE 的机制还远没这样简单，即使在绑定页面内我也没成功得到密码，因为 IE 默认并不填充密码，只有输入正确用户名后，并触发类似 onblur 事件，这个密码表单项才会填充进对应用户名的密码。这个过程我本想通过 DOM 来模拟进行的，但是没有成功。感兴趣的同学可以试试。

4. 其他浏览器

其他浏览器（除了搜狗浏览器）都和 Chrome 差不多了，大多是因为 webkit 内核。下面单独说说搜狗浏览器吧。

0x06. 搜狗浏览器“记住密码机制”的安全缺陷

搜狗浏览器在实现这个机制估计是下了一些苦工了，双核模式下都很好兼容，不过安全方面的实现存在一些问题，并没严格遵循同源策略。在我的测试中发现，搜狗没区分好不同端口及不同子域的同源问题。

比如在 www.foo.com 域下记住的密码，在 a.foo.com 与 www.foo.com:8080 域中都可以读取到。

还有一个有意思的，我们的 payload 甚至仅创建一个 password 表单项（<form>都可以不需要）就可以得到明文密码。看来搜狗浏览器在实现这个机制有偷工减料的嫌疑啊，用户体验虽然不错。

0x07. 如何防御

从三个方面进行：浏览器、网站、用户。

1. 浏览器防御

IE 的机制相对来说很不错了，其他浏览器可以借鉴，虽然这样会影响一些用户体验，我想为了更安全也值得了吧，需要特别注意的，IE 这个机制有好几个关键点，不要到时候依葫芦画瓢，学不好让人笑了:P

2. 网站防御

通常给表单的<form>标签设置 autocomplete="off"即可，不过不是所有浏览器都兼容，我发现搜狗与遨游浏览器不买这个帐。或者不要<form>标签了，通过 JS 自提交登录。新浪微博采用了这两种方式，其他网站可以学学。

3. 用户防御

意识为先吧，浏览器记住你的密码需谨慎，没必要的就不用记了。

0x08. 总结

到这还没结束，大家可以试试给表单多增加一个项或者少一个项，不同浏览器还是存在很多差异，这个大家自己找吧。

这个安全问题我很早就发现，也公开过，不过没引起足够重视:P，如果一个 SNS 类的网站中传播 XSS 蠕虫，带上这样的 payload，不知道能获取多少明文密码……或者在定点渗透过程中，如邮箱 XSS 渗透，带上这样的 payload，一定概率说不定可以拿到明文密码。怎么个危害，就看怎么个场景，怎么个利用。

最后，这篇文章的各种知识点也都会出现在我的书中（定位 Web 前端攻防），还没写完，还在继续，这里算是打个小广告了。也希望大家关注 xeye 即将上线的网站，更多分享会继续！祝各位新春快乐了:D