

JavaScript 代码混淆器

金琪琦

March 30, 2017

代码混淆的意义

代码混淆的意义

一种声音

“前端代码公开，没有秘密，本身代码就没有保护的意义。”

代码混淆的意义

一种声音

“前端代码公开，没有秘密，本身代码就没有保护的意义。”

我的观点

- 前端代码天生的不安全性决定了，应该尽可能将重要的业务代码后移动。
- 但一方面，总可能会有一些需要在前端处理，又有一定的敏感性业务；
- 另一方面，前端的一些代码往往是攻击者猜测后端漏洞的入口。

因此对于一些重要的前端代码进行适当的混淆，能够增加攻击者破译的难度。保护前端代码的同时维护整个系统的安全。

目前代码混淆在前端使用得并不多。

原因

这并不意味着前端代码不需要保护，或者对前端的代码混淆就没有意义。

而是因为前端的大多数代码并不涉及需要高安全的功能，代码混淆必然导致性能损失，对于轻量级的应用性能比安全更重要。

例子

现在越来越多网站的验证码信息不再仅仅通过一张图片，而是从前端采集用户的操作信息返回给后台判断这一系列操作是否属于人类行为。面对这样一个前端代码，一旦知道了它的采集策略就很容易伪造信息。因此对这样重要的前端代码进行混淆是很必要的。

淘宝登录代码

淘宝登录界面通过 `uab.js` 程序来采集用户信息，而这个程序就用来加载一个经过混淆的 JavaScript 程序。

合格的代码混淆器

1. 人力不可识别
2. 增加自动化还原的难度
3. 增加调试的难度
4. 性能和混淆度的均衡

实现策略

1. 代码压缩
2. 代码混淆
3. 代码防御

程序进度

1. 代码压缩

- DONE 删除注释
- DONE 删除空白符

2. 代码混淆

2.1 变量名替换

- DONE 全局变量替换为 window 的属性调用
- DONE 属性调用替换为取元素操作 []
- DONE 局部变量名随机化

2.2 常量混淆

- DONE 提取所有的字符串，通过字符数组打散
- DONE 常量编码转换

2.3 控制流替换

- TODO 将普通的循环语句展开
- TODO 将顺序执行的代码放置在精心设计的循环之中

3. 代码防御

- TODO 禁止代码格式化和变量重命名
- TODO 禁止代码调试
- TODO 域名绑定

hello

```
int i = 10;
```

遇见的问题

