**Code ObFuscation:**

**Obfuscation là gì?**

Obfuscation là 1 kĩ thuật làm cho tập lệnh khó đọc hơn nhưng vẫn hoạt động bình thường.

Đa số nhà phát triển xáo trộn code bằng các tools Obfuscator

Đoạn mã được viết bằng rất nhiều ngôn ngữ có trình thông dịch như: Python, Php, Javascript.

Trong khi Python và Php thường nằm ở phía máy chủ nên bị ẩn khỏi người dùng, còn Javascript được dùng trong những trình duyệt ở phía máy khách và đoạn mã được gửi đến người dùng và thực thi rõ ràng. Đó là lí do tại sao obfuscation thường được dùng với Javascript.

**Tại sao phải Obfuscation?**

Có rất nhiều lí do nhà phát triển phải che giấu đoạn mã của họ. Lí do chính là ẩn mã nguồn gốc và chức năng để ngăn chặn chúng khỏi bị sử dụng lại hay sao chép khi không có sự cho phép.

Một lí do nữa là cung cấp thêm 1 lớp bảo mật khi xử lí với xác thực hay mã hóa để ngăn chặn những lỗ hổng sinh ra khi lộ code.

**Chú ý là xác thực hay mã hóa ở phía máy khách là không cần thiết vì đoạn mã dễ bị tấn công hơn theo cách này.**

Tuy vậy, obfuscation cũng là 1 hành động không an toàn. Kẻ tấn công có thể làm xáo trộn đoạn mã của chúng để ngăn các hệ thống ngăn chặn và phát hiện xâm nhập phát hiện ra các tập lệnh của chúng.

**Basic ObFuscation:**

**Chạy code Javascript:**

console.log('HTB JavaScript Deobfuscation Module');

**Rút gọn code Javascript:**

Cách phổ biến để giảm khả năng đọc của 1 đoạn mã Javascript trong khi vẫn giữ đầy đủ chức năng của nó là rút gọn.

Rút gọn code là để toàn bộ mã trong một dòng (thường rất dài).

Rất nhiều công cụ có thể hỗ trợ chúng ta rút gọn code Javascript như javascript-minifier. Code Javascript rút gọn thường được lưu với phần mở rộng là .min.js

**Đóng gói mã Javascript:**

Bây giờ chúng ta sẽ xáo trộn những dòng code của ta trở nên mơ hồ và khó đọc hơn.

Hãy thử BeautifyTools

eval(function(p,a,c,k,e,d){e=function(c){return c};if(!''.replace(/^/,String)){while(c--){d[c]=k[c]||c}k=[function(e){return d[e]}];e=function(){return'\\w+'};c=1};while(c--){if(k[c]){p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('1.0(\'2 3 4 5\');',6,6,'log|console|HTB|JavaScript|Deobfuscation|Module'.split('|'),0,{}))

Bỏ vào jsconsole.com vẫn chạy bình thường.

Loại xáo trộn trên gọi là packing, được nhận biết qua 6 đối số trong 1 hàm ban đầu "function(p,a,c,k,e,d)"

Công cụ xáo trộn packer thường chuyển đổi tất cả từ và biểu tượng của code thành 1 danh sách hoặc 1 từ điển rồi sau đó tham chiếu đến nó bằng việc sử dụng (p,a,c,k,e,d) để xây lại đoạn mã cũ trong quá trình thực thi.

(p,a,c,k,e,d) có thể khác nhau từ nhà đóng gói này sang nhà đóng gói khác. Tuy vậy, nó vẫn chứa 1 thứ tự nhất định những từ và biểu tượng của mã nguồn gốc được đóng gói để biết cách sắp xếp chúng trong quá trình thực thi.

Trong khi 1 packer làm rất tốt việc giảm khả năng đọc của code, chúng ta vẫn thấy nội dung chính được viết rõ ràng, có thể tiết lộ chức năng của chúng. Đó là lí do tại sao chúng ta muốn thấy những cách làm xáo trộn code tốt hơn.

**Advance ObFuscation:**

**Obfuscator**

Copy đoạn mã trên. Trước khi nhấn obfuscate, chúng ta sẽ thay đổi String Array Encoding sang Base64. Chúng ta sẽ có đoạn mã sau

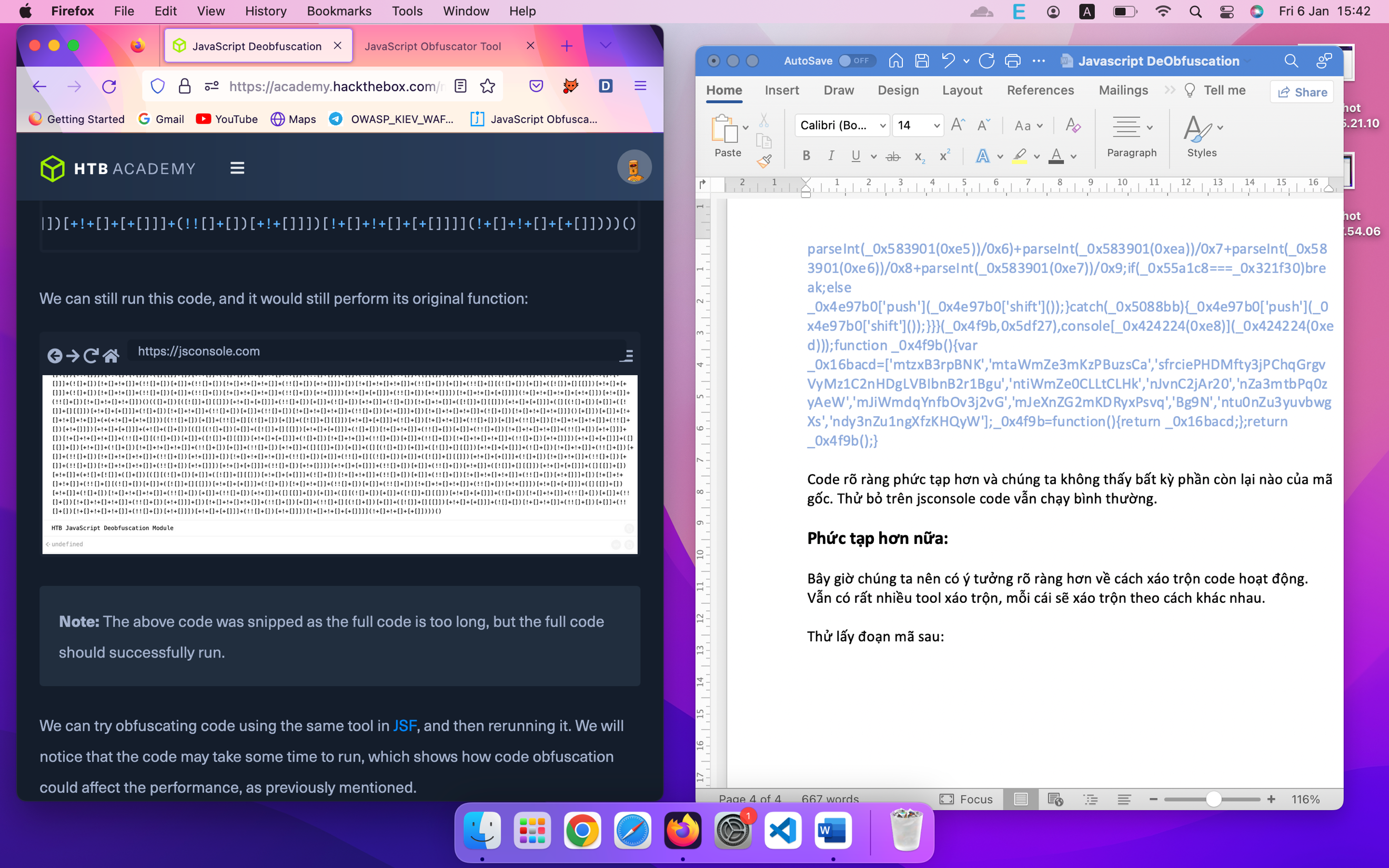
function \_0x1d59(\_0x3bfbd9,\_0x5379f4){var \_0x4f9bff=\_0x4f9b();return \_0x1d59=function(\_0x1d591e,\_0x328705){\_0x1d591e=\_0x1d591e-0xe4;var \_0x19d352=\_0x4f9bff[\_0x1d591e];if(\_0x1d59['KZbwnt']===undefined){var \_0x5bcbb0=function(\_0x1e2f8f){var \_0x520e7f='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/=';var \_0x109d48='',\_0x4b5999='';for(var \_0x26327a=0x0,\_0x591602,\_0x40ba7e,\_0x27dade=0x0;\_0x40ba7e=\_0x1e2f8f['charAt'](\_0x27dade++);~\_0x40ba7e&&(\_0x591602=\_0x26327a%0x4?\_0x591602\*0x40+\_0x40ba7e:\_0x40ba7e,\_0x26327a++%0x4)?\_0x109d48+=String['fromCharCode'](0xff&\_0x591602>>(-0x2\*\_0x26327a&0x6)):0x0){\_0x40ba7e=\_0x520e7f['indexOf'](\_0x40ba7e);}for(var \_0x14cae0=0x0,\_0x5eb3b3=\_0x109d48['length'];\_0x14cae0<\_0x5eb3b3;\_0x14cae0++){\_0x4b5999+='%'+('00'+\_0x109d48['charCodeAt'](\_0x14cae0)['toString'](0x10))['slice'](-0x2);}return decodeURIComponent(\_0x4b5999);};\_0x1d59['MWNhYZ']=\_0x5bcbb0,\_0x3bfbd9=arguments,\_0x1d59['KZbwnt']=!![];}var \_0x5a38c6=\_0x4f9bff[0x0],\_0x285eec=\_0x1d591e+\_0x5a38c6,\_0x40d3b1=\_0x3bfbd9[\_0x285eec];return!\_0x40d3b1?(\_0x19d352=\_0x1d59['MWNhYZ'](\_0x19d352),\_0x3bfbd9[\_0x285eec]=\_0x19d352):\_0x19d352=\_0x40d3b1,\_0x19d352;},\_0x1d59(\_0x3bfbd9,\_0x5379f4);}var \_0x424224=\_0x1d59;(function(\_0x5f2b0a,\_0x321f30){var \_0x583901=\_0x1d59,\_0x4e97b0=\_0x5f2b0a();while(!![]){try{var \_0x55a1c8=parseInt(\_0x583901(0xe9))/0x1+-parseInt(\_0x583901(0xec))/0x2+-parseInt(\_0x583901(0xee))/0x3\*(parseInt(\_0x583901(0xeb))/0x4)+parseInt(\_0x583901(0xe4))/0x5\*(-parseInt(\_0x583901(0xe5))/0x6)+parseInt(\_0x583901(0xea))/0x7+parseInt(\_0x583901(0xe6))/0x8+parseInt(\_0x583901(0xe7))/0x9;if(\_0x55a1c8===\_0x321f30)break;else \_0x4e97b0['push'](\_0x4e97b0['shift']());}catch(\_0x5088bb){\_0x4e97b0['push'](\_0x4e97b0['shift']());}}}(\_0x4f9b,0x5df27),console[\_0x424224(0xe8)](\_0x424224(0xed)));function \_0x4f9b(){var \_0x16bacd=['mtzxB3rpBNK','mtaWmZe3mKzPBuzsCa','sfrciePHDMfty3jPChqGrgvVyMz1C2nHDgLVBIbnB2r1Bgu','ntiWmZe0CLLtCLHk','nJvnC2jAr20','nZa3mtbPq0zyAeW','mJiWmdqYnfbOv3j2vG','mJeXnZG2mKDRyxPsvq','Bg9N','ntu0nZu3yuvbwgXs','ndy3nZu1ngXfzKHQyW'];\_0x4f9b=function(){return \_0x16bacd;};return \_0x4f9b();}

Code rõ ràng phức tạp hơn và chúng ta không thấy bất kỳ phần còn lại nào của mã gốc. Thử bỏ trên jsconsole code vẫn chạy bình thường.

**Phức tạp hơn nữa:**

Bây giờ chúng ta nên có ý tưởng rõ ràng hơn về cách xáo trộn code hoạt động. Vẫn có rất nhiều tool xáo trộn, mỗi cái sẽ xáo trộn theo cách khác nhau.

Thử lấy đoạn mã sau:



Chúng ta có thể thử xáo trộn code bằng cách dùng công cụ tương tự trong JSF và sau đó chạy lại nó.

Chúng ta chú ý thấy code sẽ mất nhiều thời gian hơn để thực thi cho thấy sự xáo trộn nó ảnh hưởng như thế nào đến hiệu suất.

Có rất nhiều công cụ xáo trộn khác như JJ Encode hoặc AA Encode. Tuy vậy, chúng thường làm code thực thi rất chậm, vì vậy không được khuyến khích dùng trừ khi có lí do rõ ràng như bỏ qua bộ lọc hay hạn chế của trang web.

**DeObfuscation:**

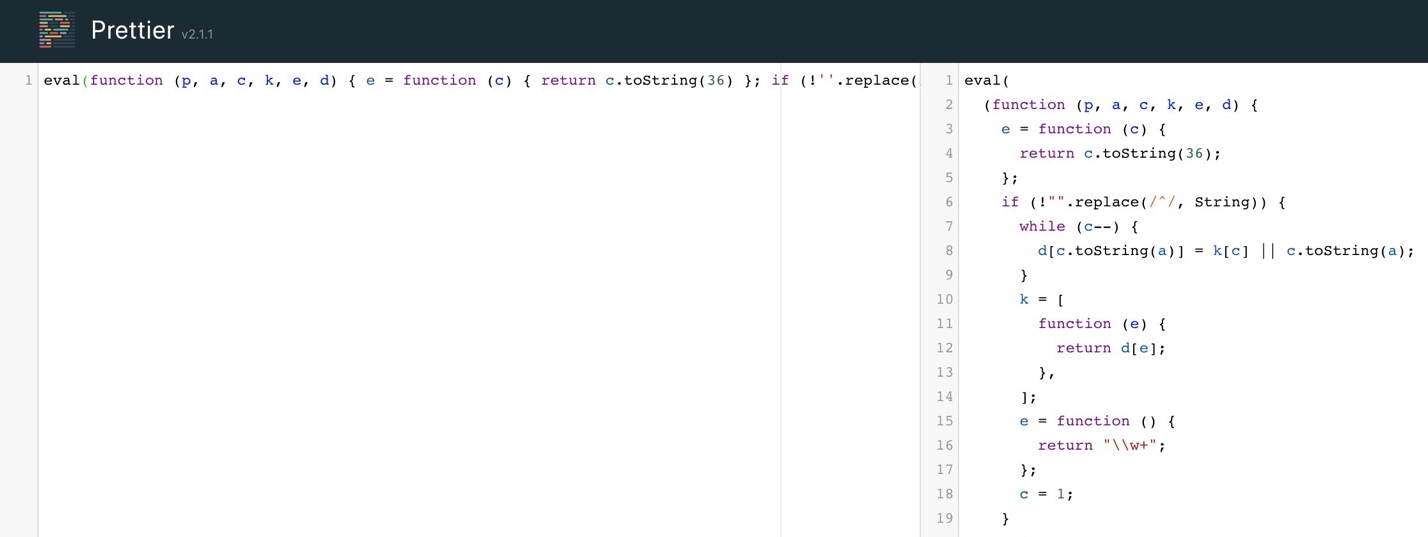
Nếu có tool xáo trộn code tự động thì cũng có những tool giải mã code tự động.

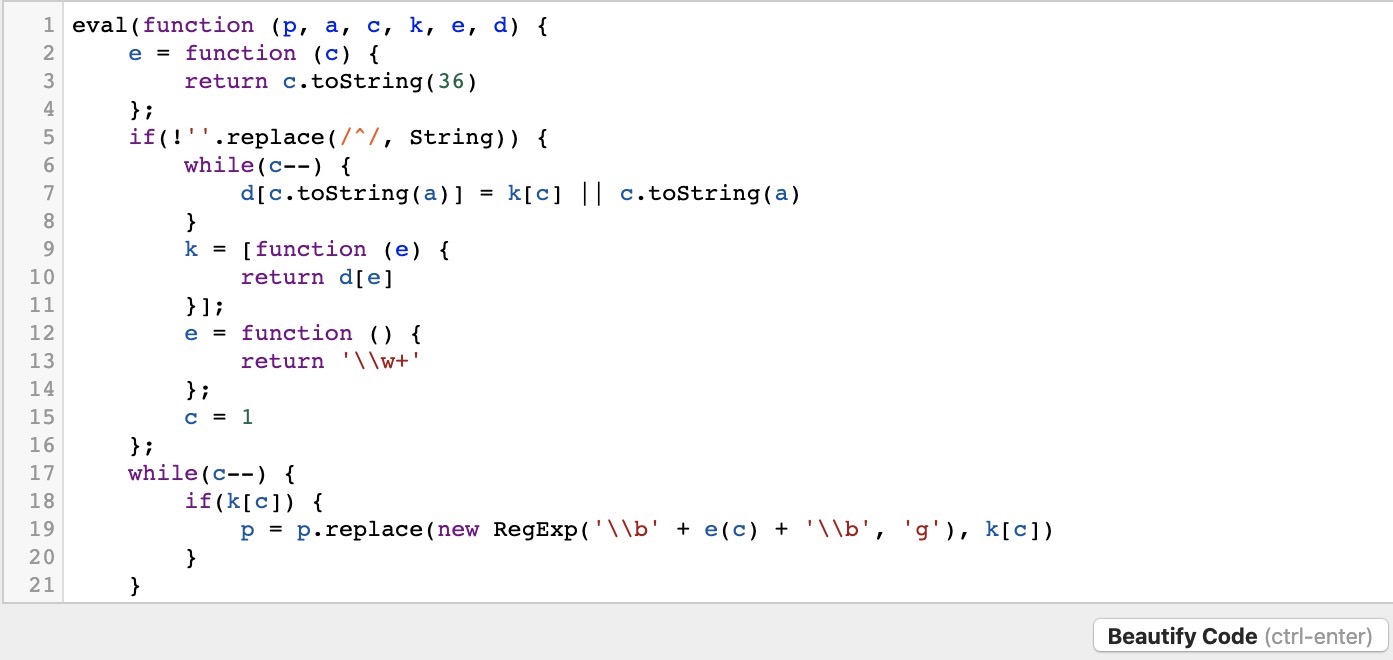
**Beautify:**

Chúng ta thấy code hiện tại chỉ viết trên 1 dòng (rút gọn code) . Để có định dạng ban đầu, chúng ta cần Beautify đoạn mã. Phương pháp cơ bản nhất là dùng Browser Dev Tools.

Ví dụ, nếu dùng Firefox, chúng ta có thể mở debugger của trình duyệt sau đó chọn vào tập lệnh secret.js. Nó sẽ hiển thị đoạn mã trong định dạng ban đầu, nhưng khi ta chọn {} ở dưới, sẽ Pretty Print tập lệnh thành định dạng Javascript phù hợp.

Hơn nữa, chúng ta có thể khởi tạo rất nhiều công cụ trực tuyến hay plugin soạn thảo code như Prettier hay Beautifier.

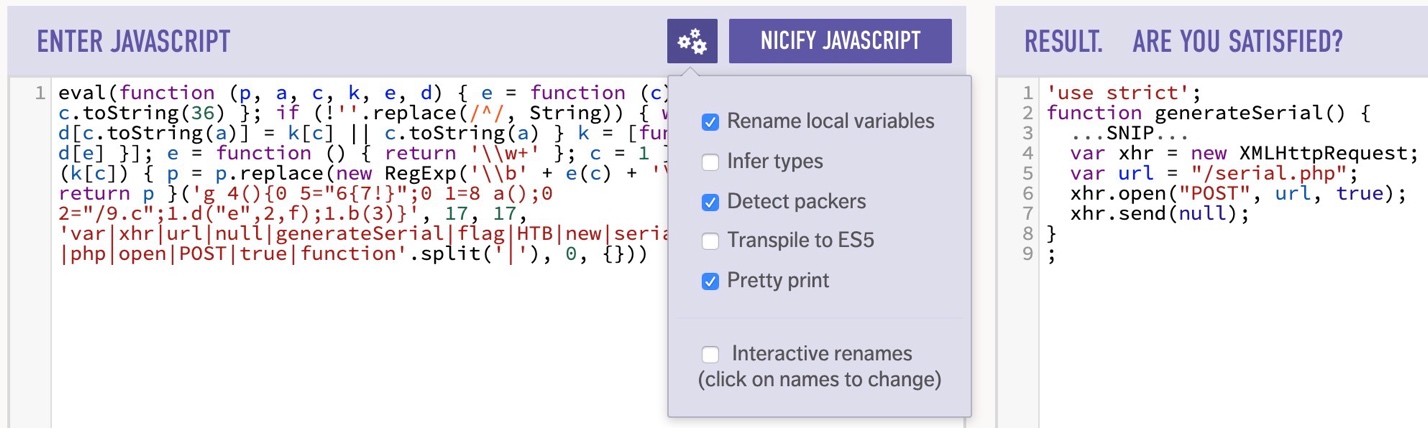




Tuy vậy code vẫn không dễ đọc. Đó là vì code chúng ta xử lí không chỉ rút gọn mà còn xáo trộn rất tốt. Vì vậy định dạng đơn giản hay làm đẹp code là không đủ. Chúng ta cần thêm công cụ để deobfuscate.

**Deobfuscate:**

Chúng ta có thể thấy rất nhiều công cụ online để giải mã code Javascript và biến nó về định dạng chúng ta có thể hiểu. Một công cụ tốt là JSNice. Thử sao chép đoạn code trên và chạy JSNice.



Chúng ta có thể tool này làm rất tốt việc giải mã code Javascript

Như đã được mô tả trước đó, phương pháp xáo trộn ở trên là packing.

Một cách khác để unpacking code như vậy là tìm giá trị trả về cuối cùng và dùng console.log để in nó thay vì thực thi.

**Reverse Engineering:**

Mặc dù những công cụ như vậy làm việc rất tốt trong việc giải mã code xáo trộn. Nhưng một khi mã trở nên phức tạp và được mã hóa nhiều hơn, nó sẽ khó khăn với những công cụ tự động.

Chúng ta sẽ cần dịch ngược đoạn mã 1 cách thủ công để hiểu cách nó được xáo trộn và chức năng cho những trường hợp này.