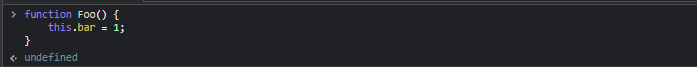
Prototype pollution

1. Một số khái niệm trong Javascript

* Một class trong Javascript được thể hiện dưới dạng function:



* Có thể coi class là template của object:

Graphical user interface

Description automatically generated with medium confidence

* Và class cũng có template là prototype, có thể truy cập qua 2 cách:

A picture containing graphical user interface

Description automatically generated

* Javascript hiển thị giá trị thuộc tính theo cách tìm ngược từ dưới lên, khi tìm kiếm foo.temp không tồn tại sẽ tìm lên foo.\_\_proto\_\_.temp, không tồn tại sẽ tìm tiếp foo.\_\_proto\_\_.\_\_proto\_\_.temp, cho tới khi gặp kết quả bằng null thì dừng lại. Cách làm này được gọi là kế thừa prototype (prototype inheritance). Ta thấy foo.\_\_proto\_\_ là prototype của class Foo, như vậy, nếu có thể thay đổi các thuộc tính trong foo.\_\_proto\_\_, thì thuộc tính trong temp sẽ thay đổi theo.

Graphical user interface, application

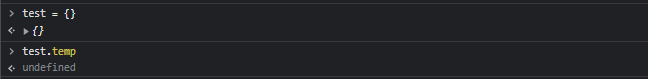
Description automatically generated with medium confidence

* Tổng hợp một chút kiến thức: khi thay đổi thuộc tính foo.\_\_proto\_\_.temp thì foo.temp cũng thay đổi, nhưng Foo.temp không thay đổi (do prototype inheritance). Để làm rõ hơn điều này, thay đổi Foo.temp và tạo đối tượng foo1 = new Foo() mới, thì foo1.temp được lấy giá trị từ foo.\_\_proto\_\_.temp như hình dưới.

Graphical user interface, application

Description automatically generated

* Tóm lại, khi thay đổi thuộc tính trong Foo.prototype sẽ thay đổi tất cả thuộc tính tương ứng của đối tượng mới trong class Foo.
* Xa hơn nữa, nếu chúng ta tạo một đối tượng rỗng mới, thuộc tính temp có tồn tại không?



* Câu trả lời là không, bởi vì foo.\_\_proto\_\_ là prototype của class Foo, còn đối tượng test không phải một đối tượng trong class Foo nên cũng không phụ thuộc vào foo.\_\_proto\_\_
* Để làm được điều này, chúng ta cần thay đổi thuộc tính trong prototype của tất cả đối tượng nói chung, ví dụ như sau:

Graphical user interface, application

Description automatically generated

* Còn để sử dụng đối tượng foo1 thì cần truy cập prototype của foo1.\_\_ptoto\_\_

Graphical user interface, application

Description automatically generated

* Đây cũng chính là nguyên lý của tấn công Prototype pollution.

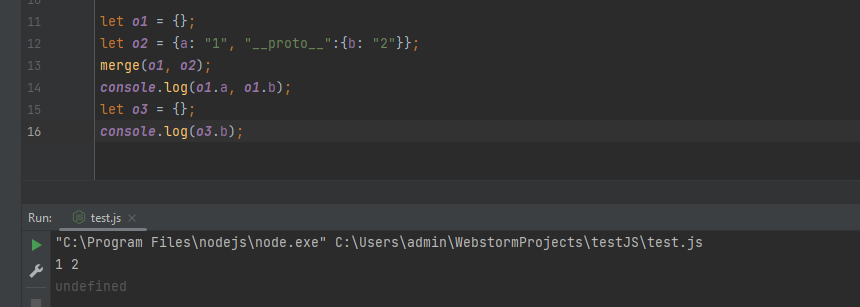
1. Khi nào lỗ hổng Prototype pollution xảy ra?

* Xem xét ví dụ sau:

Text

Description automatically generated

* Hàm merge() thực hiện hợp nhất các thuộc tính của đối tượng source vào đối tượng target, ví dụ thuộc tính a của o2 đã được hợp (thêm) vào o1.
* Xét trường hợp o2 có thuộc tính với key và value như sau:



* Thuộc tính o2.b vẫn được thêm vào đối tượng o1, nhưng khi tạo một đối tượng o3 mới thì o3.b không xác định.
* Điều này là do khi chúng ta ngay sau khi chúng ta tạo đối tượng o2, thì \_\_proto\_\_ đã tượng trưng cho prototype của o2 rồi, chứ lúc này không đóng vai trò là key nữa, dẫn đến khi chúng liệt kê các key của o2 sẽ là [a, b] nên thực chất thuộc tính trong prototype của o2 không được sửa đổi như mong muốn.

A screenshot of a computer

Description automatically generated with medium confidence

* Như vậy mục đích chúng ta cần là để \_\_proto\_\_ đóng vai trò là một key.
* JSON.parse() có thể giúp chúng ta điều này:

Graphical user interface, text, application

Description automatically generated

* JSON vẫn luôn ảo diệu như vậy! Như hình dưới thì thuộc tính o3.b đã mang giá trị 2, đồng nghĩa với việc tấn công Prototype pollution thành công!

Graphical user interface, text, application

Description automatically generated

1. Write up challenge Easy Login

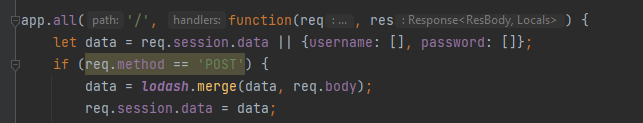
Độ khó: 200

Các chức năng: login

Lỗ hổng: Prototype pollution

Khai thác:

* Trang web sử dụng hàm lodash.merge() trong phiên bản node11 để merge data POST vào session.



* Hàm này chứa lỗ hổng Prototype Pollution.
* Điều kiện có flag là data.role nhận giá trị 1.

Text

Description automatically generated

* Thuộc tính data.role được gán bằng 1 khi đăng nhập đúng tài khoản admin (không thể), hoặc bằng 0 khi đăng nhập với guest/guest. Và nhận giá trị undefined khi không đăng nhập với 2 tài khoản này.
* Như vậy có thể thực hiện tấn công Prototype Pollution, lợi dụng cách hoạt động của cơ chế kế thừa Prototype trong Javascript, gán giá trị data.\_\_proto\_\_.role = 1, thì khi đó data.role sau khi kế thừa sẽ có giá trị bằng 1.
* Payload (lưu ý cần bỏ session cũ):

Graphical user interface, text, application

Description automatically generated