Malicious Website Detection

October 7, 2022

1 Background

This research proposed a malicious behavior suspected website detection system with static analysis and dynamic analysis based on JavaScript, Jquery, and HTML that analyzes code density, the kind of method that code used, entire JavaScript entropy, and entropy of each characteristic.

A function named myWebguard is defined. which has a constant object named 'builtins', this builtin object has property value pairs like URL:URL, parseJson:JSON.parse, apply:Function.prototype.apply, etc.

function myWebGuard () const builtins = URL: URL, Date: Date, Error: Error, Promise: Promise, setTimeout: setTimeout, parseJson: JSON.parse, stringify: JSON.stringify, window: window, sessionStorage: window.sessionStorage, apply: Function.prototype.apply, defineProperty: Object.defineProperty, hasOwnProperty: Object.prototype.hasOwnProperty, getOwnPropertyDescriptor: Object.getOwnPropertyDescriptor,

An object named Utils is created with different functions within it, it works like a key pair value, so any function with the utils can be called to run or it can be executed with the execution of utils object. There are 6 functions respectively.

1) The function() prints Mywebguard along with the argument that it takes. PrintVerbose is the key.

2)The function() with getTopOrigin as the key Returns a reference to the topmost window in the window hierarchy.it works within the try catch functionality to get the url of the current window.

const utils = printVerbose: function () console.log('[MyWebGuard]', ...arguments) , getTopOrigin: function () let url try url = builtins.window.top.origin catch url = builtins.window.origin return this.getOrigin(url) , getOrigin: function (url) try return new builtins.URL(url).hostname catch return null , isCrossOrigin: function (origin) try return this.getTopOrigin() !== origin catch return false , isUrlCrossOrigin: function (url) try return this.getTopOrigin() !== this.getOrigin(url) catch return false , sleep: function (ms) const start = new builtins.Date() let current = null do current = new builtins.Date() while (current - start ¡ ms) ,

2 JavaScript Objects and Properties

The named values, in JavaScript objects, are called properties.
Objects written as name value pairs are similar to:

Property	Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

- Associative arrays in PHP
- Dictionaries in Python
- Hash tables in C
- Hash maps in Java
- Hashes in Ruby and Perl

source: www.w3schools.com/js/$js_{o}bject_{d}efinition.asp$

JavaScript objects are dynamic "bags" of properties (referred to as own properties). JavaScript objects have a link to a prototype object. When trying to access a property of an object, the property will not only be sought on the object but on the prototype of the object, the prototype of the prototype, and so on until either a property with a matching name is found or the end of the prototype chain is reached.

Methods are actions that can be performed on objects. Object properties can be both primitive values, other objects, and functions. An object method is an object property containing a function definition.

Property	Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
FullName	function() return this.firstName + " " + this.lastName;

JavaScript objects are containers for named values, called properties and methods.

A JavaScript object is a collection of unordered properties. Properties can usually be changed, added, and deleted, but some are read only.

JavaScript objects inherit the properties of their prototype.

In JavaScript, the this keyword refers to an object.

Any JavaScript object can be converted to an array using Object.values()

Object. getOwnPropertyDescriptor

Object. defineProperty

References

[?] [?] [?] [?] [?] [?] [?] [?] [?]