Starred by 1 user

Owner:	szuend@chromium.org
CC:	yukishiino@chromium.org kenrb@chromium.org bmeu...@chromium.org yangguo@chromium.org haraken@chromium.org dsv@google.com
Status:	Fixed *(Closed)*
Components:	Platform>DevTools>JavaScript Platform>DevTools
Modified:	Apr 14, 2020
Backlog-Rank:	----
Editors:	----
EstimatedDays:	----
NextAction:	----
OS:	Linux, Windows, Chrome, Mac
Pri:	2
Type:	Bug-Security

reward-0
Security_Severity-Low
Security_Impact-Stable
allpublic
CVE_description-submitted
Target-78
Target-79
M-79
Hotlist-DevTools-CurrentSprint
Release-0-M81
CVE-2020-6447
Team-DevTools-RuntimeDebugging

Issue 991217: Security: Memory access violations when setting a breakpoint at a specific location
Reported by derce...@gmail.com on Tue, Aug 6, 2019, 11:45 AM EDT

Code

VULNERABILITY DETAILS
Using the debug console utility function, breakpoints can be set on builtin constructor functions. Internally, the devtools can call these builtin constructors as part of the process of wrapping and displaying a node object.

If a breakpoint is set on a builtin constructor function that the devtools then indirectly calls, code running within the breakpoint condition statement or while the debugger is paused has the ability to trigger various memory access violations.

VERSION
Chrome Version: Tested on 76.0.3809.87 (stable) and 78.0.3875.0 (canary)
Operating System: Windows 10 Pro, version 1903

REPRODUCTION CASE
1. The attached file forms a simple website. To begin with, download the file and place it in a directory.
2. In the directory you downloaded the file to, run the following command in a terminal:

```
python3 -m http.server 8080
```

3. In the browser, navigate to the following location:

http://localhost:8080/index.html

4. This page defines the following function:

```
function startDebugging() {
    debug(HTMLBodyElement, "var doc = new Document(); var scripts = doc.scripts;");

    console.log(document.body);
}
```

Open the devtools console and run the function:

```
startDebugging();
```

This should trigger a write access violation, crashing the renderer.

In general, it appears that there are a number of issues with code that's run while the debugger is paused. A few examples are given below. To try these, open the devtools console and run:

```
debug(HTMLBodyElement);
document.body;
```

Note that you may want to disable eager evaluation first. This is because the HTMLBodyElement constructor is only called the first time the document.body element is wrapped and displayed. If it's displayed before the debug command is run (e.g. from scrolling through past commands), nothing will happen.

When the debugger breaks, run the code snippets below via the console:

var sc = SubtleCrypto();

sc here gets set to the window object.

var win = new Window();

This should fail (with an "Illegal constructor" error). Instead it succeeds and a Window object is seemingly created. Attempting to use any of the methods results in read access violations, however.

var doc = new Document();

In general, attempting to call any of the methods on the resulting object or access its properties will result in various read access violations. The exact failure will depend on the method you call.

For example, calling:

doc.adoptNode(document.children[0]);

results in the renderer crashing within third_party\blink\renderer\platform\heap\heap_page.h. In this specific case, it appears the problem is a null pointer. In other cases, the pointer looks valid. For example, running:

var nodeName = doc.nodeName;

results in a renderer crash with a reference to a non-null address (e.g. something like 0x0000003C0A5D8CDC).

index.html
319 bytes View Download

Comment 1 by derce...@gmail.com on Tue, Aug 6, 2019, 11:51 AM EDT

I'm not sure of the exact cause of the failures here, but I can offer a few observations:

1. There's no issues if you set a breakpoint on a regular function through the debugger panel and then call that function.

2. There's also no issues if you do the same thing, except via the debug function.

3. Aside from logging document.body to the console, you can also cause the devtools to call the HTMLBodyElement constructor via the Elements panel. To do that, you would go through the following steps:

3.1. In the devtools console, run:

debug(HTMLBodyElement);

3.2. Switch to the Elements panel.
3.3. Select the body node.
3.4. On the right-hand side of the panel, select the "Event Listeners" tab.

Once you do that, the same issues occur. One thing I've noticed here is that there's no call stack shown and attempting to step into/step over the function results in a renderer crash (due to a null-pointer dereference).

I know of at least one other way to trigger this issue (where the debugger is paused in an internal function and has no call stack), but that doesn't trigger the issues above. Which suggests the issues raised above might be something specific to being paused at that particular point.

In any case, I can file a separate bug for that crash.

4. Other object types are affected. For example, Animation, Blob, Path2D, URL. Creating objects of these types and attempting to use them will result in read access violations. Other types don't seem to be affected, such as ArrayBuffer and Map.

Comment 2 by kenrb@chromium.org on Wed, Aug 7, 2019, 6:33 PM EDT Project Member

Status: Assigned (was: Unconfirmed)
Owner: bmeu...@chromium.org
Cc: kenrb@chromium.org
Labels: OS-Chrome OS-Linux OS-Mac OS-Windows Pri-1
Components: Platform>DevTools Platform>DevTools>JavaScript

Assigning to bmeurer@ for assessment.

I don't know if there is anything that can be done here, or if it is significant. A user adding a breakpoint to a builtin that runs other code would not normally be considered within Chrome's threat model.

Comment 3 by bmeu...@chromium.org on Thu, Aug 8, 2019, 12:24 AM EDT Project Member

Owner: yangguo@chromium.org
Cc: bmeu...@chromium.org
Labels: Security_Severity-Low

Comment 4 by yangguo@chromium.org on Thu, Aug 8, 2019, 4:16 AM EDT Project Member

Can reproduce. Stack trace:

```
#0 0x55eafca36b19 base::debug::CollectStackTrace()
#1 0x55eafc994883 base::debug::StackTrace::StackTrace()
#2 0x55eafca366a1 base::debug::(anonymous namespace)::StackDumpSignalHandler()
#3 0x7fec2c88b3a0 <unknown>
#4 0x55eaff8e9876 blink::Node::CreateRareData()
#5 0x55eaff845fef blink::ContainerNode::EnsureNodeLists()
#6 0x55eaff8691db blink::Document::scripts()
#7 0x55eaff35b769 blink::V8Document::ScriptsAttributeGetterCallback()
#8 0x55eafb8f97e6 v8::internal::FunctionCallbackArguments::Call()
#9 0x55eafb8f8d28 v8::internal::(anonymous namespace)::HandleApiCallHelper<>()
#10 0x55eafb8f8719 v8::internal::Builtins::InvokeApiFunction()
#11 0x55eafbba3b1a v8::internal::Object::GetPropertyWithAccessor()
#12 0x55eafbba359d v8::internal::Object::GetProperty()
#13 0x55eafbcafb05 v8::internal::Runtime::GetObjectProperty()
#14 0x55eafbcb50e2 v8::internal::Runtime_GetProperty()
#15 0x55eafc165b59 <unknown>
  r8: 000008242a57de20  r9: 0000000000000001 r10: 0026783900af4738 r11: 0000000000000001
 r12: 000030d35bb804b1 r13: 00007fffb6b35738 r14: 000030d35bb804e9 r15: 000055eaff35b6f0
  di: 000008242a31d880  si: 0000000000000000  bp: 00007fffb6b355b0  bx: 000030d35bb804b1
  dx: 0000000000a7efd8  ax: 00002858f82d0af0  cx: 0000000000000000  sp: 00007fffb6b355a0
  ip: 000055eaff8e9876 efl: 0000000000010206 cgf: 002b000000000033 erf: 0000000000000007
 trp: 000000000000000e msk: 0000000000000000 cr2: 000030d35bb804e9
[end of stack trace]
Calling _exit(1). Core file will not be generated.
```

Comment 5 by derce...@gmail.com on Thu, Aug 8, 2019, 8:14 AM EDT

Re c#2:

I'm not sure if it is possible to take advantage of the issues here, but if it is, the general idea would be that a script on the page would run the necessary code in the breakpoint condition statement (passed to the debug function). The script would still need the ability to call the debug function, and for that the user would need to type something into the console.

I think I can also explain part of the cause of the issues here:

IsValidConstructorMode is called to check whether a constructor function can be used to create a new object:

https://cs.chromium.org/chromium/src/third_party/blink/renderer/platform/bindings/v8_object_constructor.cc?l=57&rcl=498f5876beea9d297f7844a22ec935265645c692

Typically, ConstructorMode::Current(info.GetIsolate()) will return ConstructorMode::kCreateNewObject and IsValidConstructorMode will throw an exception.

However, because the script is paused in V8ObjectConstructor::NewInstance, the constructor mode is set to ConstructorMode::kWrapExistingObject:

https://cs.chromium.org/chromium/src/third_party/blink/renderer/platform/bindings/v8_object_constructor.cc?l=44&rcl=498f5876beea9d297f7844a22ec935265645c692

That means that when you call something like:

new Window()

IsValidConstructorMode won't throw and the call will return a value.

Comment 6 by yangguo@chromium.org on Thu, Aug 8, 2019, 9:26 AM EDT Project Member

Given this code:

```
function startDebugging() {
    debug(HTMLBodyElement, "var doc = new Document(); var scripts = doc.scripts;");
    console.log(document.body);
}
```

We get this crash stack trace:

```
#0 0x7fdbb1f92d9f base::debug::CollectStackTrace()
#1 0x7fdbb1ccb73d base::debug::StackTrace::StackTrace()
#2 0x7fdbb1ccb6f8 base::debug::StackTrace::StackTrace()
#3 0x7fdbb1d1aca9 logging::LogMessage::~LogMessage()
#4 0x7fdb8affe4be blink::ReportFatalErrorInMainThread()
#5 0x7fdb841de6f0 v8::Utils::ReportApiFailure()
#6 0x7fdb8421bd7e v8::Object::SlowGetAlignedPointerFromInternalField()
#7 0x7fdb8aecb422 blink::GetInternalField<>()
#8 0x7fdb8aecb2fd blink::ToScriptWrappable()
#9 0x7fdb8afc57fd blink::V8Document::ToImpl()
#10 0x7fdb8d53b7bb blink::document_v8_internal::ScriptsAttributeGetter()
#11 0x7fdb8d53b75f blink::V8Document::ScriptsAttributeGetterCallback()
#12 0x7fdb842beabe v8::internal::FunctionCallbackArguments::Call()
#13 0x7fdb842bd291 v8::internal::(anonymous namespace)::HandleApiCallHelper<>()
#14 0x7fdb842bbe9f v8::internal::Builtins::InvokeApiFunction()
#15 0x7fdb847e46a5 v8::internal::Object::GetPropertyWithAccessor()
#16 0x7fdb847e38d0 v8::internal::Object::GetProperty()
#17 0x7fdb8499a105 v8::internal::Runtime::GetObjectProperty()
#18 0x7fdb849a52e2 v8::internal::__RT_impl_Runtime_GetProperty()
#19 0x7fdb852aa820 <unknown>
```

The issue is roughly this:

- We set a breakpoint at the constructor for HTMLBodyElement.
- Accessing document.body invokes said constructor
- The debug break is a conditional, so we evaluate the condition.
- In the condition, we try to get the "script" property.
- This property is implemented as accessor.
- The accessor callback in Blink opens up the object wrapper, and accesses the internal fields to figure out the type of the object.
- These internal fields are actually the undefined value, so they are not aligned pointers.
- In debug mode, this crashes with a check failure.
- In release mode, we dereference a field in the undefined oddball object and crash.

Comment 7 by yangguo@chromium.org on Thu, Aug 8, 2019, 9:30 AM EDT Project Member

Cc: haraken@chromium.org

Comment 8 by yangguo@chromium.org on Thu, Aug 8, 2019, 9:31 AM EDT Project Member

Note that the crash only reproduces if V8 is patched with this:

```
diff --git a/src/debug/debug-evaluate.cc b/src/debug/debug-evaluate.cc
index 0d8a7b2c7e..614fb0e03f 100644
--- a/src/debug/debug-evaluate.cc
+++ b/src/debug/debug-evaluate.cc
@@ -102,10 +102,12 @@ MaybeHandle<Object> DebugEvaluate::WithTopmostArguments(Isolate* isolate,

   // Materialize receiver.
   Handle<String> this_str = factory->this_string();
-  JSObject::SetOwnPropertyIgnoreAttributes(
-      materialized, this_str, Handle<Object>(it.frame()->receiver(), isolate),
-      NONE)
-      .Check();
+  if (!it.frame()->receiver().IsTheHole(isolate)) {
+    JSObject::SetOwnPropertyIgnoreAttributes(
+        materialized, this_str, Handle<Object>(it.frame()->receiver(), isolate),
+        NONE)
+        .Check();
+  }

   // Use extension object in a debug-evaluate scope.
```

```
  Handle<ScopeInfo> scope_info =
```

Otherwise we run into another failing assertion first :)

Comment 9 by haraken@chromium.org on Thu, Aug 8, 2019, 10:16 AM EDT Project Member

Cc: yukishiino@chromium.org

> - In the condition, we try to get the "script" property.
> - This property is implemented as accessor.
> - The accessor callback in Blink opens up the object wrapper, and accesses the internal fields to figure out the type of the object.

Here the "object wrapper" means the wrapper of Document (not HTMLBodyElement), right? The wrapper of Document should have been constructed at that point. It looks strange that you hit an undefined field there.

HTMLBodyElement may be in a half-baked state but doc.scripts will never access the wrapper of HTMLBodyElement.

+yukishiino for more insights.

Comment 10 by sheriffbot@chromium.org on Thu, Aug 8, 2019, 10:19 AM EDT Project Member

Labels: -Pri-1 Pri-2

Setting Pri-2 to match security severity Low. If this is incorrect, please reset the priority. Sheriffbot won't make this change again.

For more details visit https://www.chromium.org/issue-tracking/autotriage - Your friendly Sheriffbot

Comment 11 by bugdroid on Fri, Aug 9, 2019, 8:46 AM EDT Project Member

The following revision refers to this bug:
 https://chromium.googlesource.com/v8/v8.git/+/7d81b0517ce70997c3ca29e5b70b9693e33ae8e3

commit 7d81b0517ce70997c3ca29e5b70b9693e33ae8e3
Author: Yang Guo <yangguo@chromium.org>
Date: Fri Aug 09 12:46:15 2019

[debugger] ignore receiver for construct frames for evaluate

R=szuend@chromium.org

~~Bug: chromium:991217~~
Change-Id: Icf4d5522fe2a1d2400e6dd33744d6a60ab4e634c
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/1745469
Commit-Queue: Yang Guo <yangguo@chromium.org>
Reviewed-by: Simon Zünd <szuend@chromium.org>
Cr-Commit-Position: refs/heads/master@{#63144}

[modify] https://crrev.com/7d81b0517ce70997c3ca29e5b70b9693e33ae8e3/src/debug/debug-evaluate.cc
[modify] https://crrev.com/7d81b0517ce70997c3ca29e5b70b9693e33ae8e3/test/cctest/test-debug.cc

Comment 12 by yukishiino@chromium.org on Fri, Aug 9, 2019, 10:01 AM EDT Project Member

#c5 and #c6 are totally correct.

Blink's ConstructorMode must be kCreateNewObject whenever author script is running.
https://cs.chromium.org/chromium/src/third_party/blink/renderer/platform/bindings/v8_object_constructor.h?rcl=46f63a50f247408a186263b9f5a4e3db211cd264&l=45

Since the debugger interrupted the execution during creation of V8 wrapper object, the constructor mode is left as kWrapExistingObject when interrupted.

At this point, you can run arbitrary author script while the constructor mode is kWrapExistingObject (= NOT kCreateNewObject).

Re #c9(haraken): The "object wrapper" is a wrapper of a newly created Document, which is half-baked. The newly created document is not associated with a Blink object because it's created while kWrapExistingObject. The newly created document is different from the main document of the page ("var doc = new Document").

The root cause is that Blink is assuming that author script never runs while creating a new V8 wrapper object. However, the debugger interrupted it and made it possible to run author script in the middle of wrapper creation.

Comment 13 by haraken@chromium.org on Sat, Aug 10, 2019, 9:34 PM EDT Project Member

Thanks shiino-san! Your analysis sounds very reasonable :)

Comment 14 by yukishiino@chromium.org on Tue, Aug 13, 2019, 5:54 AM EDT Project Member

Hi yangguo@, what do you think of the following idea?

I'm thinking of something like v8::DebugForbiddenScope in which scope all breakpoints stop interrupting the execution. Blink will use the scope like:

```
    v8::DebugForbiddenScope dont_run_author_script_scope;
    v8::Local<v8::Object> wrapper = CreateV8Wrapper();
```

In the case of the original issue report, debug(HTMLBodyElement) is supposed to interrupt the execution of HTMLBodyElement kicked by web author. It's not supposed to interrupt a wrapper object creation.

Is v8::DebugForbiddenScope a feasible solution? What do you think?

Comment 15 by yangguo@chromium.org on Tue, Aug 13, 2019, 4:16 PM EDT Project Member

With that scope, the above example code will no longer trigger a breakpoint, is that correct? To trigger a breakpoint, we would beed to explicitly use `new HTMLBodyElement`, right?

It sounds like a feasible solution to me. I'm just wondering how many of those wrapper creation sites we need to put into such a scope. Do you have a clue whether it's easy to find all these places in blink?

Comment 16 by yukishiino@chromium.org on Tue, Aug 13, 2019, 11:47 PM EDT Project Member

> With that scope, the above example code will no longer trigger a breakpoint, is that correct? To trigger a breakpoint, we would beed to explicitly use `new HTMLBodyElement`, right?

Yes.

> I'm just wondering how many of those wrapper creation sites we need to put into such a scope. Do you have a clue whether it's easy to find all these places in blink?

https://cs.chromium.org/chromium/src/third_party/blink/renderer/platform/bindings/v8_dom_wrapper.cc?rcl=1cd23288a0118331ab4ed8f2215958cf642469d1&l=40

I think V8DOMWrapper::CreateWrapper must be the only place that needs the scope. All wrapper creation sites end to calling V8DOMWrapper::CreateWrapper, and V8DOMWrapper::CreateWrapper is responsible for V8 wrapper creation.

Comment 17 by yukishiino@chromium.org on Mon, Aug 19, 2019, 3:01 AM EDT Project Member

yangguo@, would you support the scope on V8 side, if the idea works well (or assign someone else)?

Comment 18 by yangguo@chromium.org on Mon, Aug 19, 2019, 3:31 AM EDT Project Member

I can do it. I'm just currently thinking about whether there is a better way for us to detect this class of invocations to ignore for debug breaks.

Comment 19 by yukishiino@chromium.org on Mon, Aug 19, 2019, 5:01 AM EDT Project Member

Thanks! If there'd be a better way, I'm happier, too.

Comment 20 by mmoroz@google.com on Tue, Aug 20, 2019, 1:03 AM EDT Project Member

Labels: Security_Impact-Stable M-77

Comment 21 by bugdroid on Thu, Aug 22, 2019, 5:06 AM EDT Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8.git/+/e66cee7e9e9630bd8ee28943a3ca0da20fed647d

commit e66cee7e9e9630bd8ee28943a3ca0da20fed647d
Author: Yang Guo <yangguo@chromium.org>
Date: Thu Aug 22 09:01:56 2019

[debug] only break on entry when immediately called from JS

When we break on function entry, check whether the target function is being
called from JS after entering V8 through V8's API. We implement this by
keeping track of the stack height when we enter V8 through the API, and compare
the caller JS frame's stack height with that.

R=szuend@chromium.org

~~Bug: chromium:991217~~, ~~chromium:992406~~
Change-Id: I258ad9cef11fe0ef48de6fd5055790792fd0ec0c
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/1762298
Commit-Queue: Yang Guo <yangguo@chromium.org>
Reviewed-by: Simon Zünd <szuend@chromium.org>
Cr-Commit-Position: refs/heads/master@{#63331}

[modify] https://crrev.com/e66cee7e9e9630bd8ee28943a3ca0da20fed647d/include/v8.h
[modify] https://crrev.com/e66cee7e9e9630bd8ee28943a3ca0da20fed647d/src/api/api.cc
[modify] https://crrev.com/e66cee7e9e9630bd8ee28943a3ca0da20fed647d/src/api/api.h
[modify] https://crrev.com/e66cee7e9e9630bd8ee28943a3ca0da20fed647d/src/debug/debug.h
[modify] https://crrev.com/e66cee7e9e9630bd8ee28943a3ca0da20fed647d/src/execution/isolate.cc
[modify] https://crrev.com/e66cee7e9e9630bd8ee28943a3ca0da20fed647d/src/execution/thread-local-top.cc
[modify] https://crrev.com/e66cee7e9e9630bd8ee28943a3ca0da20fed647d/src/execution/thread-local-top.h
[modify] https://crrev.com/e66cee7e9e9630bd8ee28943a3ca0da20fed647d/src/runtime/runtime-debug.cc
[modify] https://crrev.com/e66cee7e9e9630bd8ee28943a3ca0da20fed647d/test/cctest/test-debug.cc

Comment 22 by yangguo@chromium.org on Thu, Aug 22, 2019, 5:36 AM EDT Project Member

Status: Fixed (was: Assigned)

Comment 23 by yukishiino@chromium.org on Thu, Aug 22, 2019, 8:59 AM EDT Project Member

Thanks for the fix!

Comment 24 by bugdroid on Thu, Aug 22, 2019, 9:33 AM EDT Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8.git/+/0bd19ddbba48c6eea759d24532cb9a95def09049

commit 0bd19ddbba48c6eea759d24532cb9a95def09049
Author: Maya Lekova <mslekova@chromium.org>
Date: Thu Aug 22 13:29:59 2019

Revert "[debug] only break on entry when immediately called from JS"

This reverts commit e66cee7e9e9630bd8ee28943a3ca0da20fed647d.

Reason for revert: Speculative revert for https://ci.chromium.org/p/chromium/builders/try/linux-rel/173349

Original change's description:
> [debug] only break on entry when immediately called from JS
>
> When we break on function entry, check whether the target function is being
> called from JS after entering V8 through V8's API. We implement this by
> keeping track of the stack height when we enter V8 through the API, and compare
> the caller JS frame's stack height with that.
>
> R=szuend@chromium.org
>
> ~~Bug: chromium:991217~~, ~~chromium:992406~~
> Change-Id: I258ad9cef11fe0ef48de6fd5055790792fd0ec0c
> Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/1762298
> Commit-Queue: Yang Guo <yangguo@chromium.org>
> Reviewed-by: Simon Zünd <szuend@chromium.org>
> Cr-Commit-Position: refs/heads/master@{#63331}

TBR=yangguo@chromium.org,szuend@chromium.org

Change-Id: I4bfb42f7ce1484807696048a09609f14113d10f4
No-Presubmit: true
No-Tree-Checks: true
No-Try: true
~~Bug: chromium:991217~~, ~~chromium:992406~~
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/1762525
Reviewed-by: Maya Lekova <mslekova@chromium.org>
Commit-Queue: Maya Lekova <mslekova@chromium.org>
Cr-Commit-Position: refs/heads/master@{#63341}

[modify] https://crrev.com/0bd19ddbba48c6eea759d24532cb9a95def09049/include/v8.h
[modify] https://crrev.com/0bd19ddbba48c6eea759d24532cb9a95def09049/src/api/api.cc
[modify] https://crrev.com/0bd19ddbba48c6eea759d24532cb9a95def09049/src/api/api.h
[modify] https://crrev.com/0bd19ddbba48c6eea759d24532cb9a95def09049/src/debug/debug.h

[modify] https://crrev.com/0bd19ddbba48c6eea759d24532cb9a95def09049/src/execution/isolate.cc
[modify] https://crrev.com/0bd19ddbba48c6eea759d24532cb9a95def09049/src/execution/thread-local-top.cc
[modify] https://crrev.com/0bd19ddbba48c6eea759d24532cb9a95def09049/src/execution/thread-local-top.h
[modify] https://crrev.com/0bd19ddbba48c6eea759d24532cb9a95def09049/src/runtime/runtime-debug.cc
[modify] https://crrev.com/0bd19ddbba48c6eea759d24532cb9a95def09049/test/cctest/test-debug.cc

Comment 25 by sheriffbot@chromium.org on Thu, Aug 22, 2019, 10:31 AM EDT Project Member

Labels: -Restrict-View-SecurityTeam Restrict-View-SecurityNotify

Comment 26 by natashapabrai@google.com on Mon, Aug 26, 2019, 12:01 PM EDT Project Member

Labels: reward-topanel

Comment 27 by jdeblasio@chromium.org on Wed, Oct 2, 2019, 6:58 PM EDT Project Member

Status: Started (was: Fixed)
Labels: -Restrict-View-SecurityNotify -reward-topanel Restrict-View-SecurityTeam

yangguo@: can you take another look at this? This should have been re-opened when your change was reverted.

Comment 28 by sheriffbot@chromium.org on Wed, Oct 23, 2019, 9:11 AM EDT Project Member

Labels: -M-77 Target-78 M-78

Comment 29 by sheriffbot@chromium.org on Wed, Dec 11, 2019, 9:12 AM EST Project Member

Labels: -M-78 Target-79 M-79

Comment 30 by mea...@chromium.org on Mon, Jan 6, 2020, 8:01 PM EST Project Member

yangguo: Pinging, can you please take another look?

Comment 31 by bmeu...@chromium.org on Tue, Jan 7, 2020, 4:11 AM EST Project Member

Status: Assigned (was: Started)
Owner: szuend@chromium.org
Cc: yangguo@chromium.org

Simon, can you take a look while Yang is away?

Comment 32 by bmeu...@chromium.org on Tue, Jan 7, 2020, 4:11 AM EST Project Member

Labels: Hotlist-DevTools-CurrentSprint

Comment 33 by szuend@chromium.org on Tue, Jan 7, 2020, 5:19 AM EST Project Member

Status: Fixed (was: Assigned)

The change from Yang was relanded as https://crrev.com/c/1767996 and I manually verified that the code is in. This bug didn't get updated because the reland CL description is missing the bug number.

Closing the bug since Yang marked it as fixed with the initial CL.

Comment 34 by mea...@chromium.org on Tue, Jan 7, 2020, 1:14 PM EST Project Member

Great, thank you!

Comment 35 by natashapabrai@google.com on Tue, Jan 14, 2020, 11:57 AM EST Project Member

Labels: reward-topanel

Comment 36 by natashapabrai@google.com on Wed, Jan 29, 2020, 7:04 PM EST Project Member

Labels: -reward-topanel reward-0

Unfortunately the Panel declined to reward this report

Comment 37 by adetaylor@google.com on Fri, Mar 13, 2020, 2:02 PM EDT Project Member

Labels: Release-0-M81

The commit in #c33 is definitely in Chrome M81. It may also be in older Chrome versions, but I'm going to credit this in the M81 release notes.

Comment 38 by adetaylor@chromium.org on Fri, Mar 13, 2020, 2:32 PM EDT Project Member

Labels: CVE-2020-6447 CVE_description-missing

Comment 39 by sheriffbot on Tue, Apr 14, 2020, 1:56 PM EDT Project Member

Labels: -Restrict-View-SecurityTeam allpublic

This bug has been closed for more than 14 weeks. Removing security view restrictions.

For more details visit https://www.chromium.org/issue-tracking/autotriage - Your friendly Sheriffbot

Comment 40 by adetaylor@chromium.org on Tue, Apr 14, 2020, 3:14 PM EDT Project Member

Labels: -CVE_description-missing CVE_description-submitted