Show navigation

Temporarily disabling escape analysis

Published 22 September 2017 · tagged with security

In JavaScript, an allocated object *escapes* if it is accessible from outside the current function. Normally V8 allocates new objects on the JavaScript heap, but using *escape analysis*, an optimizing compiler can figure out when an object can be treated specially because its lifetime is provably bound to the function's activation. When the reference to a newly allocated object does not escape the function that creates it, JavaScript engines don't need to explicitly allocate that object on the heap. They can instead effectively treat the values of the object as local variables to the function. That in turn enables all kinds of optimizations like storing these values on the stack or in registers, or in some cases, optimizing the values away completely. Objects that escape (more accurately, objects that can't be proven to not escape) must be heap-allocated.

For example, escape analysis enables V8 to effectively rewrite the following code:

```
function foo(a, b) {
  const object = { a, b };
  return object.a + object.b;
  // Note: `object` does not escape.
}
```

...into this code, which enables several under-the-hood optimizations:

```
function foo(a, b) {
  const object_a = a;
  const object_b = b;
  return object_a + object_b;
}
```

V8 v6.1 and older used an escape analysis implementation that was complex and generated many bugs since its introduction. This implementation has since been removed and a brand new escape analysis codebase is available in V8 v6.2.

However, a Chrome security vulnerability involving the old escape analysis implementation in V8 v6.1 has been discovered and responsibly disclosed to us. To protect our users, we've turned off escape analysis in Chrome 61. Node.js should not be affected as the exploit depends on execution of untrusted JavaScript.

Turning off escape analysis negatively impacts performance because it disables the abovementioned optimizations. Specifically, the following ES2015 features might suffer temporary slowdowns:

- destructuring
- `for-of` iteration
- array spread
- rest parameters

Note that disabling escape analysis is only a temporary measure. With Chrome 62, we'll ship the brand new — and most importantly, enabled — implementation of escape analysis as seen in V8 v6.2.

Posted by Mathias Bynens (@mathias), sandbox escape analyzer.

Retweet this article!