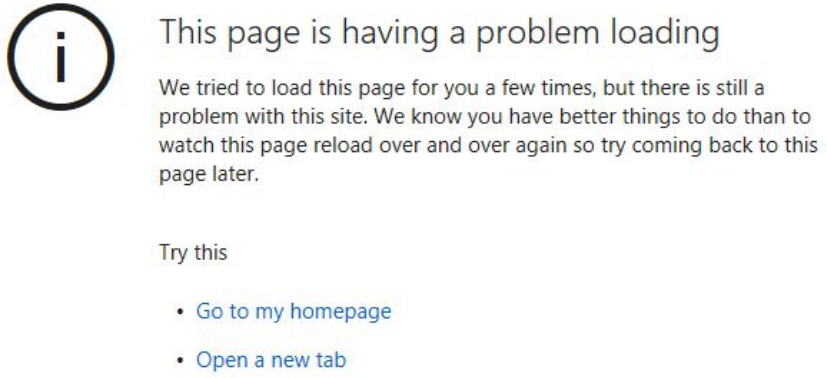
Google

The Problems and Promise of WebAssembly

About Me

- Natalie Silvanovich AKA natashenka
- Project Zero member
- Previously did mobile security on Android and BlackBerry
- Defensive-turned-offensive researcher



What is WebAssembly?

- Format for writing assembly-like code in JavaScript
- Motivated by need for greater efficiency and safety
- Compilability is a major goal
- WC3 standard
- Applications beyond browsers

What *Is* WebAssembly

- WebAssembly starts as a binary
  - ArrayBuffer or TypedArray
  - Can load using fetch (or not)

```
var wasm = new Uint8Array(123);
wasm[0] = 0x0;
wasm[1] = 0x61;
wasm[2] = 0x73;
wasm[3] = 0x6d;
wasm[4] = 0x1;
wasm[5] = 0x0;
wasm[6] = 0x0;
wasm[7] = 0x0;
wasm[8] = 0x1;
wasm[9] = 0xa;
```

WebAssembly Binary Format

- Consists of sections of various types (some optional)
- Mandatory order and duplicates forbidden

Field	Type	Description
magic number	`uint32`	Magic number `0x6d736100` (i.e., '\0asm')
version	`uint32`	Version number, `0x1`

section

Field	Type	Description
id	`varuint7`	section code
payload_len	`varuint32`	size of this section in bytes
name_len	`varuint32` ?	length of `name` in bytes, present if `id == 0`
name	`bytes` ?	section name: valid UTF-8 byte sequence, present if `id == 0`
payload_data	`bytes`	content of this section, of length `payload_len - sizeof(name) - sizeof(name_len)`

Section Types

Section Name	Code	Description
Type	`1`	Function signature declarations
Import	`2`	Import declarations
Function	`3`	Function declarations
Table	`4`	Indirect function table and other tables
Memory	`5`	Memory attributes
Global	`6`	Global declarations
Export	`7`	Exports
Start	`8`	Start function declaration
Element	`9`	Elements section
Code	`10`	Function bodies (code)
Data	`11`	Data segments

WebAssembly Module

- First step is parsing binary format and loading it into Module

```
var m = new WebAssembly.Module(wasm);
```

What could go wrong?

CVE-2018-4121 -- WebKit: WebAssembly parsing does not correctly check section order

- Order check can be bypassed

```
static inline bool validateOrder(Section previous, Section next)
{
    if (previous == Section::Custom)
        return true;
    return static_cast<uint8_t>(previous) < static_cast<uint8_t>(next);
}
```

What could go wrong?

CVE-2018-6092 -- V8: Integer Overflow when Processing WebAssembly Locals

- Integer overflow

```
if ((count + type_list->size()) > kV8MaxWasmFunctionLocals)
{
        decoder->error(decoder->pc() - 1, "local count too
large");
        return false;
      }
```

What could go wrong?

CVE-2018-4222 -- WebKit: Info leak in WebAssembly Compilation

- Can read out of bounds of the wasm buffer

```
var b2 = new ArrayBuffer(1000);
var view = new Int8Array(b2, 700);
var mod = new WebAssembly.Module(a);
```

WebAssembly Instance

- Loads module into runnable form
  - Loads imports
  - Initializes imports
  - Creates exports

WebAssembly Imports

- Three import types
  - Function: JavaScript or WebAssembly function
  - Memory: memory page object
  - Table: function table object
- If two wasm Modules have the same Memory and Table, they are in the same compartment
- There is no practical reason for a Module to share one of these objects but not the other

WebAssembly Memory

- Memory page for WebAssembly code
- Has a initial and max size, and can be expanded by calling grow in WebAssembly or JavaScript
- Accessed by WebAssembly instructions

```
var memory = new WebAssembly.Memory({initial:10, maximum:100});
memory.grow(10);
```

What could go wrong?

- Overflows in expanding Memory
  - CVE-2018-5093 -- FireFox: Buffer overflow in WebAssembly during Memory/Table resizing (found by OSS-Fuzz)
  - CVE-2017-15399 -- V8: UaF in Growing Memory (Zhao Qixun of Qihoo 360 Vulcan Team)

What could go wrong?

- Surprisingly few OOB issues
  - Limited and known set of WebAssembly instructions
  - Limited threading
  - Safe signal buffers

Tables

- Function table for WebAssembly
- Can only contain WebAssembly functions
- Only need to set at startup in practice, but can be changed any time
- Can grow similar to a Memory page

```
var t = new WebAssembly.Table({initial:2, element:"anyfunc"});
```

What could go wrong?

- Overflows in expanding Table
  - CVE-2018-5093 -- Buffer overflow in WebAssembly during Memory/Table resizing (found by OSS-Fuzz)
  - CVE-2017-5122: OOB access in v8 wasm after Symbol.toPrimitive overwrite (found by Choongwoo Han of Naver Corporation working with Chromium Vulnerability Rewards)

Initialization

- Data segments from WebAssembly binary are used to initialize Memory
- Element segments from WebAssembly binary are use to initialize Elements

What could go wrong?

- No OOB issues seen so far!
- V8: 826434: UaF in Calling Table
  - If a table is changed during a call to a function in the table, there is a UaF, as it drops the handle to its instance
  - Fixed by preventing table change during call
  - Still possible due to element initialization

Exports

- End result of creating Module and then creating an Instance is exported WebAssembly functions ready to call!

```
var mod = new WebAssembly.Module(wasm);
var i = new WebAssembly.Instance(mod,
        {imported : {func : f}, js : {table : t, mem : m} });

i.exports.exported_func(); // WebAssembly happens!
```

Runtime Issues?

- Instructions do wrong thing*
- Incorrect bounds checking
- Incorrect handles / UaF

Future Issues

- Concurrency
- WebAssembly-GC

Conclusion

- Several vulnerabilities have been found in WebAssembly implementations
- WebAssembly has features that make vulnerabilities less likely
- The future direction of WebAssembly features will determine its security

Questions and Discussion

http://googleprojectzero.blogspot.com/

@natashenka
natashenka@google.com