



Bento

a software MMU for WebAssembly

Fine-Grained Memory Isolation for COTS WebAssembly Binaries

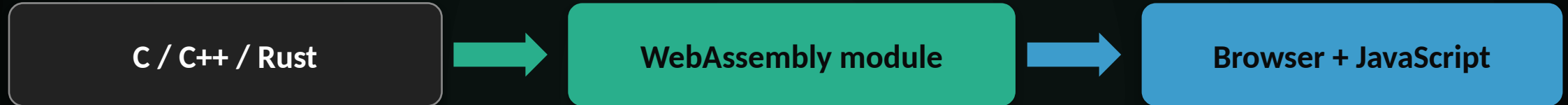
Oussama Draissi Lucas Davi

paluno - University of Duisburg-Essen

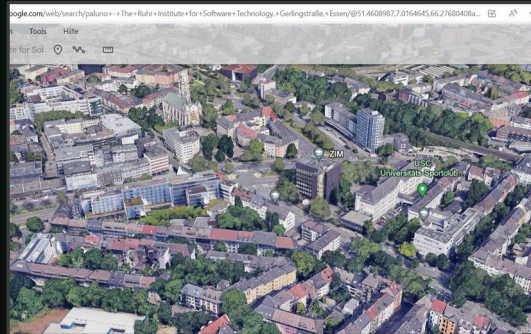
The Web Conference 2026 · Dubai, UAE · July 2026

WebAssembly Powers the Web

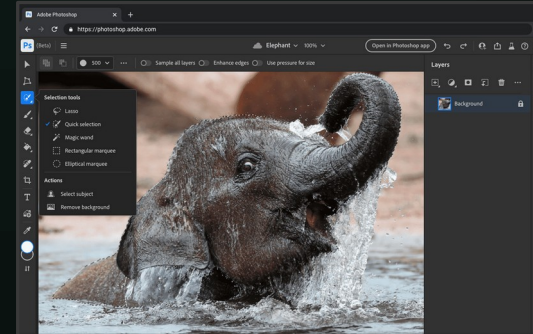
[Haas et al., PLDI 2017]



Twitch



Google Maps



Photoshop



Zoom

5%+

of all Chrome page loads instantiate a Wasm module

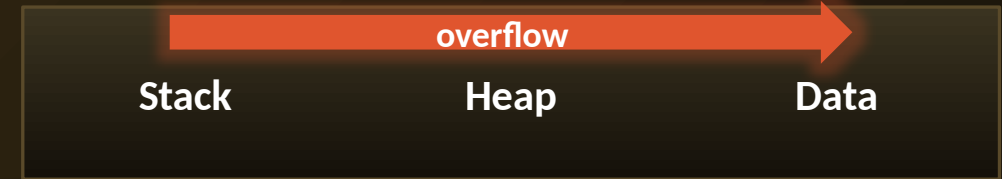
Security by Design

Wasm's design removes classic memory exploits. One gap remains.

SOUND BY DESIGN

- ✓ Code is immutable, so no code injection
- ✓ Structured control flow gives CFI for free
- ✓ Sandboxed execution is SFI by default
- ✓ DEP, stack canaries and ASLR are unneeded

THE GAP



- Linear memory has no internal isolation
- Data-only corruption still crosses regions

Control-flow defenses do not stop **data-only attacks inside one flat memory.**

From a Memory Error to XSS

CuteCats.com decodes a **third-party image** in the visitor's browser. One crafted file is enough.



A bug in one region rewrites trusted data in another.
No control-flow hijack, **just a full cross-site scripting attack.**

This Threat Is Everywhere

Measured in our [Wemby](#) paper · Draissi et al., ISSTA 2025

OUR PRIOR WORK

37,000

domains analyzed

2,782

domains vulnerable

< 2%

dataflows sanitized

A prevalent, real-world threat model.

Related Work

No prior defense protects **deployed Wasm**.

Need source, runtime, or hardware

PKUWA

[Lei et al., CCS'23]

RLBox

[Narayan et al., USENIX Sec'20]

MS-Wasm

[Michael et al., POPL'23]

WaVe

[Johnson et al., S&P'23]

Fragile, need debug symbols

Fuzzm

[Lehmann et al., '21]

WBSan

[Wu et al., WWW'25]

detect bugs after the fact, never prevent them

The open gap: **proactive hardening for stripped COTS Wasm, on the standard web.**

THE SOLUTION

Every prior defense needs **more** than a stripped binary has.



Bento

Give each region of a Wasm binary **its own isolated memory**.

binary-only

no source, no symbols

automatic

no developer effort

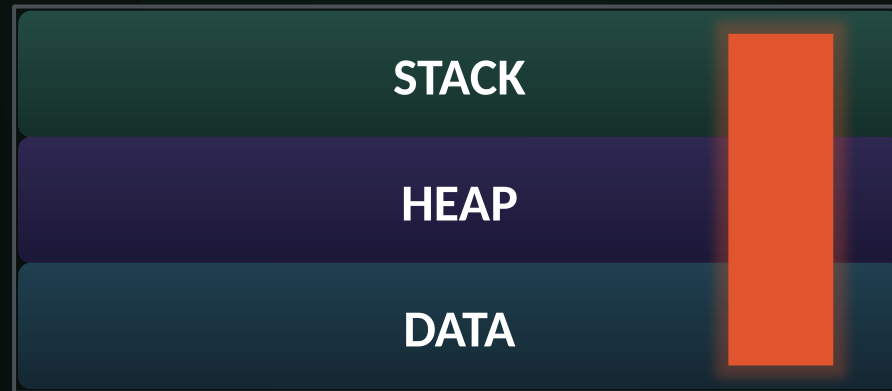
near-zero overhead

no deployment penalty

| Today: One Shared Memory

a normal load into the one shared memory

```
i32.load $image
```



Stack, heap and data share **one linear memory**.

Bento: Isolated Memories

the same load, now bound to a separate memory

```
i32.load @Heap $image
```

Wasm multi-memory feature · standard since Wasm 3.0

@Heap is just a memory index. Zero new instructions.

STACK
memory 1

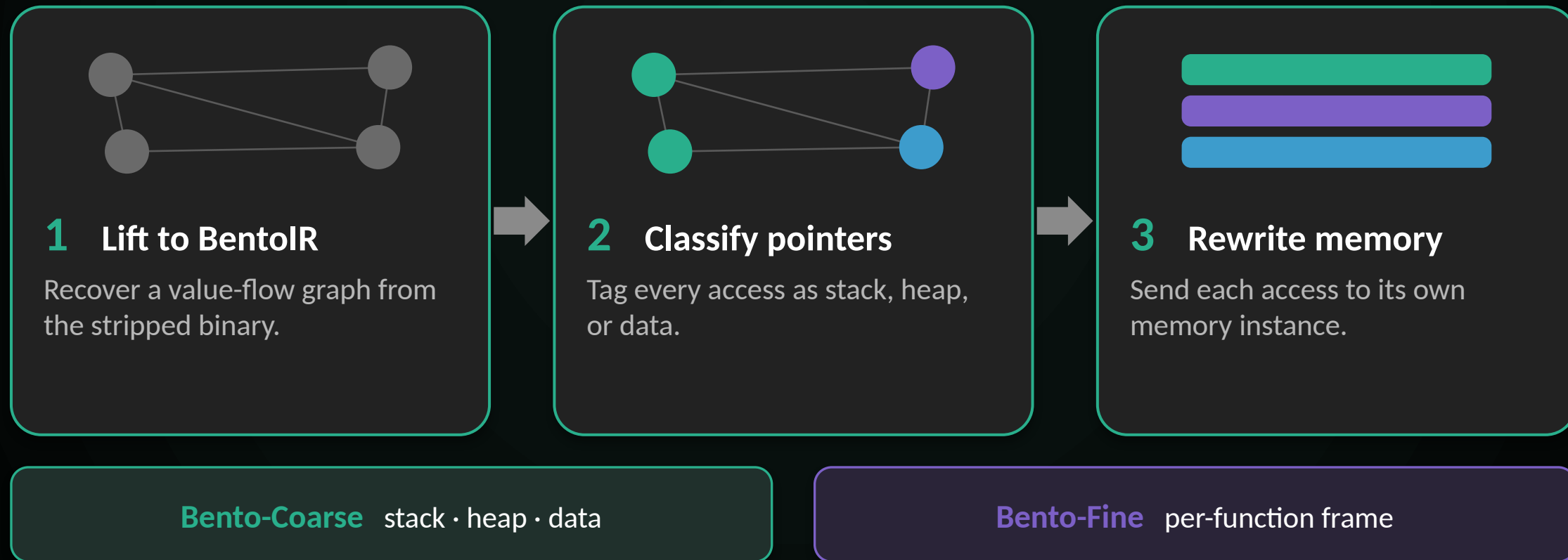
HEAP
memory 2

DATA
memory 0

Each region is its own memory instance, **isolated by the runtime.**

The index is baked into immutable code, so a bug cannot redirect it.

How Bento Works



Fully automatic, binary to binary. **The output is a standard Wasm module.**

Bento Stops the Exploit

The same malicious image hits CuteCats.com. Libpng CVE-2018-14550.

WITHOUT BENTO

image buffer



HTML tag

alert("XSS!") runs in CuteCats

WITH BENTO

image buffer



HTML tag **safe**



CuteCats loads.
No XSS.

The overflow stays in the buffer's own memory. **The HTML tag is untouched.**

Evaluation

We harden and differentially test **every binary**, across three optimization levels.

THE CORPUS

52

WABench + real-world apps

300

csmith programs

3

real-world CVEs

DOMAINS scientific computing · cryptography · image processing · NLP · databases · networking

× 3 optimization levels -00 · -02 · -03



1,065

unique stripped binaries

WHAT WE FOUND

100%

identical memory traces

every access matches the unmodified baseline

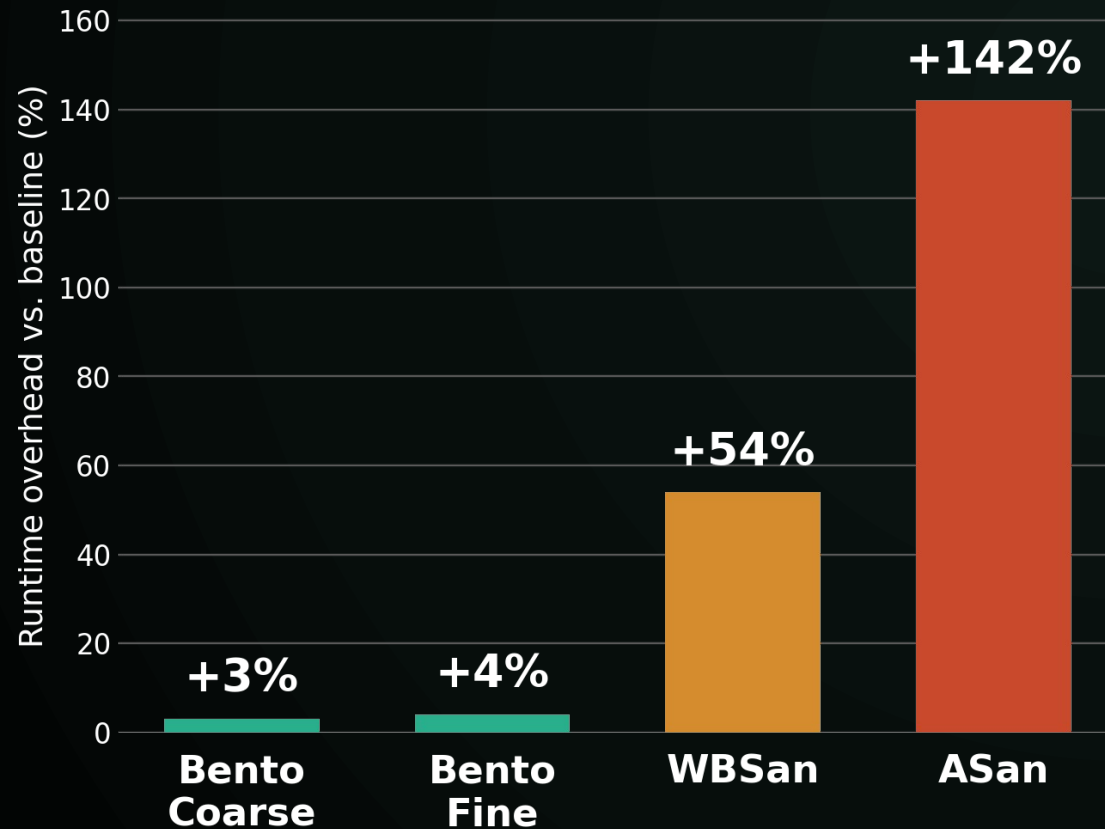
100%

hardened automatically

no debug symbols, no manual steps — even custom allocators (libpng) detected

Differential testing across 1,065 stripped binaries confirms Bento is **semantically sound**.

I Near-Zero Overhead



Zero instructions added

1.00× fuel, no computational cost

Runs on stripped COTS

where WBSan and ASan fail

Architectural hardening: **3 to 4% runtime**, **~1% startup and memory**, automatic on stripped COTS.

Bento at a Glance

This Threat Is Everywhere

Measured in our [Wemby](#) paper · Draissi et al., ISSTA 2025

OUR PRIOR WORK

37,000

domains analyzed

2,782

domains vulnerable

< 2%

dataflows sanitized

A prevalent, real-world threat model.

MOTIVATION

The threat

Bento: Isolated Memories

the same load, now bound to a separate memory

```
i32.load @Heap $image
```

Wasm multi-memory feature · standard since Wasm 3.0

@Heap is just a memory index. Zero new instructions.

STACK
memory 1

HEAP
memory 2

DATA
memory 0

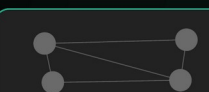
Each region is its own memory instance, isolated by the runtime.

The index is baked into immutable code, so a bug cannot redirect it.

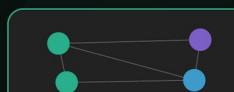
KEY IDEA

The idea

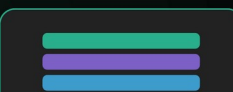
How Bento Works



1 Lift to BentoIR
Recover a value-flow graph from the stripped binary.



2 Classify pointers
Tag every access as stack, heap, or data.



3 Rewrite memory
Send each access to its own memory instance.

Bento-Coarse stack · heap · data

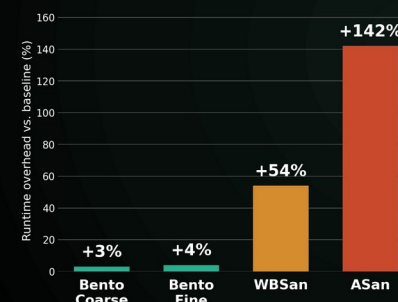
Bento-Fine per-function frame

Fully automatic, binary to binary. The output is a standard Wasm module.

DESIGN

How it works

Evaluation: Near-Zero Overhead



Zero instructions added
1.00x fuel, no computational cost

100% identical traces
across 1065 stripped binaries

Runs on stripped COTS
where WBSan and ASan fail

Architectural hardening: 3 to 4% runtime, ~1% startup and memory, automatic on stripped COTS.

RESULTS

The results

SUMMARY

Bento

Thank You

Questions and collaboration are very welcome



scan to visit 0drai.xyz



EMAIL

oussama.draissi@uni-due.de



LINKEDIN

linkedin.com/in/oussama-draissi



WEBSITE

0drai.xyz