ARMore

Pushing Love Back Into Binaries

Luca Di Bartolomeo, Hossein Moghaddas, Mathias Payer
Late stage code modifications

Binary rewriting allows late-stage code modifications preserving original functionality.

Common use cases include:

1. Hardening (CFI)
2. Profiling (Valgrind)
3. Translation (QEMU)
4. Fuzzing (AFL-QEMU)
Challenge 1: Distinguishing code and data

```
movz x0, 0x10
add x0, x0, 0x20
...
.data:
.string "sneaky string!!"
... 
... 
ldr x0, [.data]
b puts
```

Any mistake is fatal!
Challenge 2: Pointer construction

Aarch64 uses 4-byte fixed ISA, but pointers are 64 bit
Requires multiple instructions to “construct” a pointer:

```
adrp x0, 0x8000
str x0, [sp, -0x8]
div x1, x2, x4
br x3
ldr x0, [sp, -0x8]
add x0, 0x128
```

```
adrp x0, 0x8000
add x0, x0, 0x128 . . .
add x0, 0x128
adrp x0, 0x8000
str x0, [sp, -0x8]
div x1, x2, x4
br x3
ldr x0, [sp, -0x8]
add x0, 0x128
```

```
adrp x0, 0x8000
mov x1, x0
add x1, x1, 0x128
```

How to recover the value of a pointer and rewrite it to preserve its target?

Previous approaches relied on heuristics to rewrite pointers!
ARMore for non-PIC code: Layout replication

How to distinguish data and pointers?

Replicate exactly the same address space layout.

Pointers don’t need to be adjusted anymore: they will point to the correct data by construction.

No need to distinguish pointers from data anymore!
ARMCore for PIC code: Pointer construction

But what about PIC?

On aarch64 **only two instructions** can read the program counter register:

- `bl` / `blr` (Branch and link)
- `adrp` (Address page)

Every single pointer construction will **always** start with an `adrp`

PIC is handled by making all `adrp` target the replicated layout
We modify our layout replication to introduce the **rebound table**:

```
.section rebound_table
0x400: b .text+0x0
0x404: b .text+0x4
0x408: b .text+0x8
0x40c: b .text+0x20
0x410: b .text+0x24
0x414: b .text+0x28
```

Transparent translation of code pointers at the cost of a single branch!
New feature on ARM 8.2:

XOM: Execute-only memory

1. Set .text permissions to “--x”
2. Install a segfault handler only for .text read violations
3. Keep an old copy of .text and return the correct value

Support of data mixed with text without heuristics!
ARMore use-case: Fuzzing!

ARMore comes with batteries included:

Coverage instrumentation to fuzz closed-source software at the same speeds as if you had source code. (3x faster than AFL-QEMU!)

Binary Address Sanitizer instrumentation makes triaging crashes easier than ever!

2 CVEs on closed source Nvidia software for CUDA
ARMore: Spread the Love for Aarch64 rewriting

Main challenges for Aarch64 rewriting:

- Distinguishing code and data
- Recovering pointer constructions

Key takeways:

- Binary rewriting for Aarch64 is easier and more precise than x86.
- No need for heuristics to rewrite aarch64 binaries!
- ARMore is open source at: https://github.com/HexHive/RetroWrite