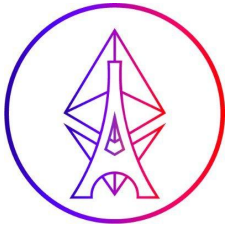


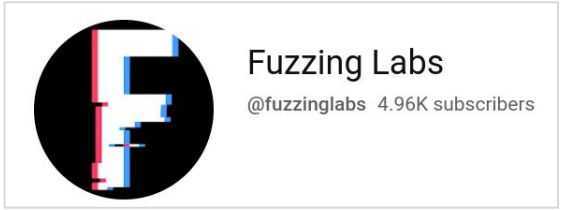
FUZZING LABS

Journey into Building Security Tools

For Cairo/Starknet Smart Contracts

Patrick Ventuzelo (@Pat_Ventuzelo)

- Founder & CEO of **FuzzingLabs** | Senior **Security Researcher**
  - Fuzzing and vulnerability research
    - **100+ bugs founds in L1/L2 protocols**
  - Development of security tools
    - **thoth**, cairo-fuzzer, octopus, etc.



- Training/Online courses
  - **Rust** Security Audit & Fuzzing
  - **Go** Security Audit & Fuzzing
  - **WebAssembly** Reversing & Analysis
  - **Ethereum/Solidity** Security (WIP)
  - **Cairo/Starknet** Security (WIP)

- Blockchain security since 2016
  - Public Speaker: EthCC speaker (x3), Devcon, DSS
  - Research about EVM reversing & Tx analysis
  - Lead developer of Beaconfuzz, beacon chain differential fuzzer
  - **Fuzzing and audits** of dozen of L1/L2 protocol implementations
  - Currently building a **web3 profiling & deanonymization tool**

Thoth,
the Cairo/Starknet security toolkit

Compilation - Cairo into JSON artifact

```
%builtins output

from starkware.cairo.common.alloc import alloc
from starkware.cairo.common.serialize import serialize_word

func array_sum(arr : felt*, size) -> (sum):
    if size == 0:
        return (sum=0)
    end

    let (sum_of_rest) = array_sum(arr=arr + 1, size=size - 1)
    return (sum=[arr] + sum_of_rest)
end

func main{output_ptr : felt*}():
    const ARRAY_SIZE = 3

    # Allocate an array.
    let (ptr) = alloc()

    assert [ptr] = 9
    assert [ptr + 1] = 16
    assert [ptr + 2] = 25

    let (sum) = array_sum(arr=ptr, size=ARRAY_SIZE)

    serialize_word(sum)

    return ()
end
```

```
"data": [
    "0x40780017fff7fff",
    "0x1",
    "0x208b7fff7fff7ffe",
    "0x400380007ffc7ffd",
    "0x482680017ffc8000",
    "0x1",
    "0x208b7fff7fff7ffe",
    "0x20780017fff7ffd",
    "0x5",
    "0x480680017fff8000",
    "0x0",
    "0x208b7fff7fff7ffe",
    "0x482680017ffc8000",
    "0x1",
    "0x482680017ffd8000",
    "0x800000000000110000000000000000000000
    "0x1104800180018000",
    "0x800000000000010ffffffffffffffffffff
    "0x480280007ffc8000",
    "0x48307ffe7fff8000",
    "0x208b7fff7fff7ffe",
    "0x1104800180018000",
```

Why Thoth has been created?

- Thoth, the Cairo/Starknet security toolkit - github

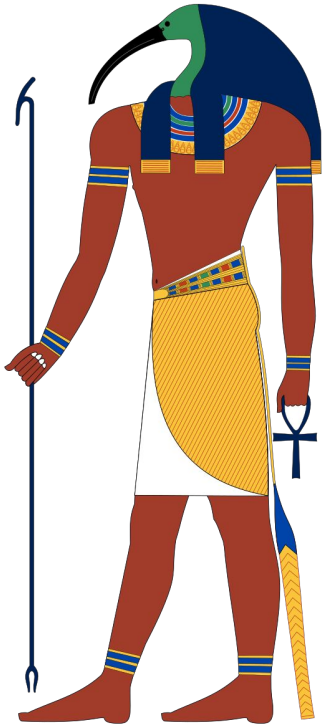
- Problematic
  - Most contracts on the mainnet/testnet are **not verified**
  - **Only** the full **JSON artifact** is mandatory and stored online
- Goal
  - Analysis of closed source contract for due diligence
  - Help developers to understand compiler operations
  - Thoth is intended to be a complete tool

SOURCE CODE STATUS

Not Verified

- Name: **Thoth**
  - God of the **moon**, **sacred texts**, mathematics, sciences, magic, messenger and recorder of the deities, **master of knowledge**, and patron of scribes.
  - Inspired by other amazing tools
    - Octopus, Slither, Mythril, etc.

- **Only the bytecode is the Truth**



Thoth - Evolution

Thoth 0.1.0 - Disassembler

- *"**Visual representation** of the bytecode as a linear **sequence of instructions**."*

- Several data are in the **JSON**.
- **Interesting information**
  - Builtins
  - Structures
  - Events
  - Constants representation
  - Functions ID and names matching

```
"data": [
    "0x40780017fff7fff",
    "0x1",
    "0x208b7fff7fff7ffe",
    "0x400380007ffc7ffd",
    "0x482680017ffc8000",
    "0x1",
    "0x208b7fff7fff7ffe",
    "0x20780017fff7ffd",
    "0x5",
    "0x480680017fff8000",
    "0x0"
```

```
        func starkware.cairo.common.alloc.alloc{}() -> (ptr : felt*)

offset 0:           NOP
offset 0:           ADD                 AP, 1               # memory[ap] = segments.add()
offset 2:           RET

        func starkware.cairo.common.serialize.serialize_word{output_ptr : felt*}(word : felt)

offset 3:           ASSERT_EQ           [FP-3], [[FP-4]]
offset 4:           ASSERT_EQ           [AP], [FP-4] + 1
offset 4:           ADD                 AP, 1
offset 6:           RET

        func __main__.array_sum{}(arr : felt*, size : felt) -> (sum : felt)

offset 7:           JNZ                 5
offset 9:           ASSERT_EQ           [AP], 0
offset 9:           ADD                 AP, 1
offset 11:          RET
offset 12:          ASSERT_EQ           [AP], [FP-4] + 1
offset 12:          ADD                 AP, 1
offset 14:          ASSERT_EQ           [AP], [FP-3] + -1
offset 14:          ADD                 AP, 1
offset 16:          CALL                7                   # __main__.array_sum
offset 16:          ADD                 AP, 2
offset 18:          ASSERT_EQ           [AP], [[FP-4]]
offset 18:          ADD                 AP, 1
offset 19:          ASSERT_EQ           [AP], [AP-1] + [AP-2]
offset 19:          ADD                 AP, 1
offset 20:          RET
```

Thoth 0.2.0 - Decompiler

- *"A decompiler is a computer program that **takes bytecode** as input, and **attempts to create a high-level source file** that (ideally) can be successfully compiled."*

- Features
  - Recovery of parameters from function calls.
  - Generation of imports.

- The first version of the decompiler
  - Similar to the disassembly output.
  - AP/FP is **complicated** to understand **for beginners.**
  - Limited support of **if/else** blocks.

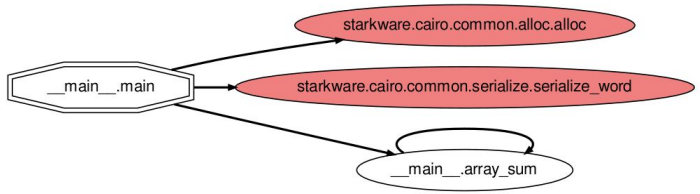
```
func __main__.array_sum{}(arr : felt*, size : felt) -> (sum : felt)
    if [AP-3] == 0:
        # 0 -> 0x0
        [AP] = 0;         ap ++
        return([ap-1])

    end
    [AP] = [FP-4] + 1;    ap ++
    [AP] = [FP-3] + -1;    ap ++
    array_sum([ap-2], [ap-1])
    [AP] = [[FP-4]];    ap ++
    [AP] =  [AP-1] + [AP-2];    ap ++
    return([ap-1])
end

func __main__.main{output_ptr : felt*}()
    alloc()
    # 9 -> 0x9
    [AP] = 9;    ap ++
    [AP-1] = [[AP-2]]
    # 16 -> 0x10
    [AP] = 16;    ap ++
    [AP-1] = [[AP-3]+1]
    # 25 -> 0x19
    [AP] = 25;    ap ++
    [AP-1] = [[AP-4]+2]
    [AP] = [AP-4];    ap ++
    # 3 -> 0x3
    [AP] = 3;    ap ++
    array_sum([ap-2], [ap-1])
    [AP] = [FP-3];    ap ++
    [AP] = [AP-2];    ap ++
    serialize_word([ap-2], [ap-1])
    ret
end
```

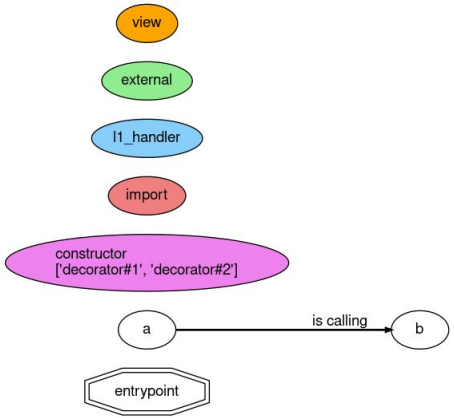
Thoth 0.2.0 - Call Graph

- *"Call graph represents calling **relationships between subroutines** in a computer program."*

- **Node** represents a function.
- **Edge(a, b)** indicates that function **a** calls function **b**.

- Legend:
  - Colors for important functions (import, constructor, etc.)
  - Octagonal shape for **entry-point**.



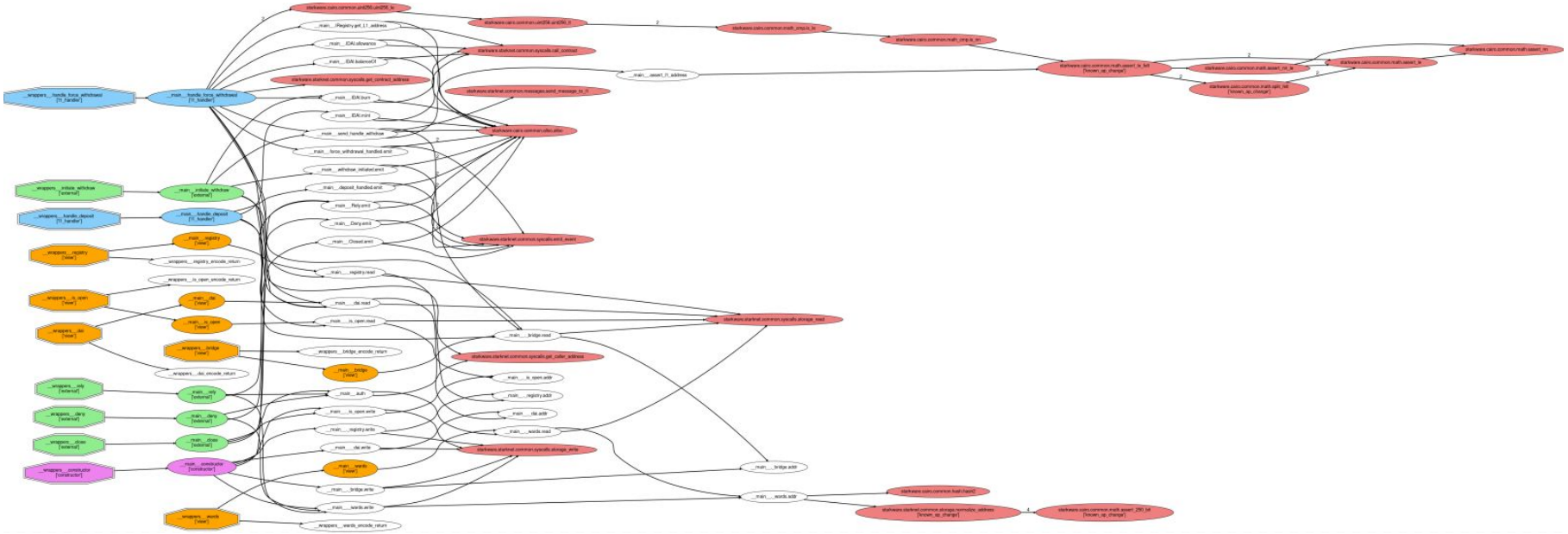



You Retweeted

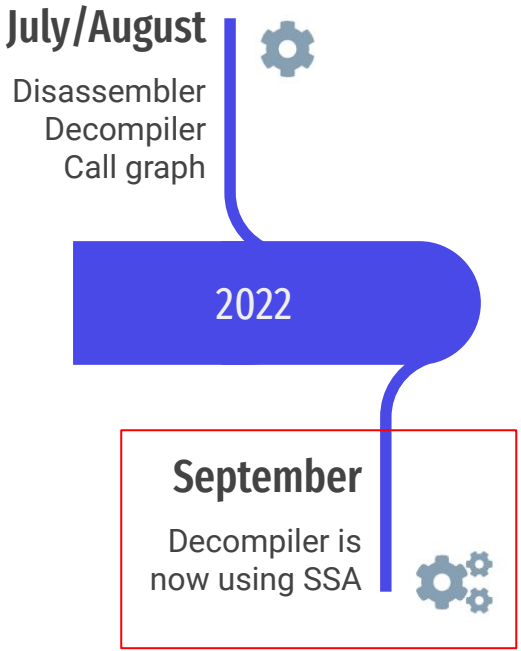
Carbonable.io @Carbonable_io · Oct 19

/3 To schematize the architecture of our contracts with their dependencies, we also used Thoth to generate the callgraphs of our functions proposed by @FuzzingLabs :

Thoth 0.2.0 - Advanced example (dai bridge)



Thoth - Evolution



Thoth 0.3.0 - SSA Decompiler

- Major decompilation improvement
  - By leveraging on the **CFG**.
  - Introduction of **Single Static Assignment (SSA)**.
  - Creation of a **virtual stack of variable** per basic block.

- Single Static Assignment (SSA)
  - *"**Static single assignment** form (abbreviated **SSA** form/SSA) is a property of an intermediate representation (IR), which requires that **each variable is assigned exactly once**, and every variable is defined before it is used."*
  - Each variable is assigned once.
  - Each variable is defined before being used.
  - **phi node (Φ)** represents multiple potential value for a same variable chosen depending on the predecessor of the current block.

```
// Function 0
func __main__.main{}(){
    v0 = 3    // 0x3
    assert v1 = 47    // /
    if (v1 == 0) {
        v2 = v1
    else:
        v3 = 2    // 0x2
    }
    v4 = 50    // 2
    v5 = Φ(v2, v3) + 3
    assert v4 = v6 + v5
    if (v6 == 0) {
        v7 = v6
    else:
        v8 = 2    // 0x2
    }
    assert v9 = Φ(v7, v8) - 53
    if (v9 == 0) {
        v10 = 25    // 0x19
    else:
        v11 = 2    // 0x2
    }
    assert v12 = Φ(v10, v11) + 47
    if (v12 == 0) {
        v13 = 25    // 0x19
    else:
        v14 = 2    // 0x2
    }
```

Thoth 0.3.0 - Decompiler evolution

- Original Source code

```
func main{}():
    let a = 1
    let b = 2
    let c = 3
    myfunc(b, a, c)
    myfuncbis(c, a)
    ret
end
```

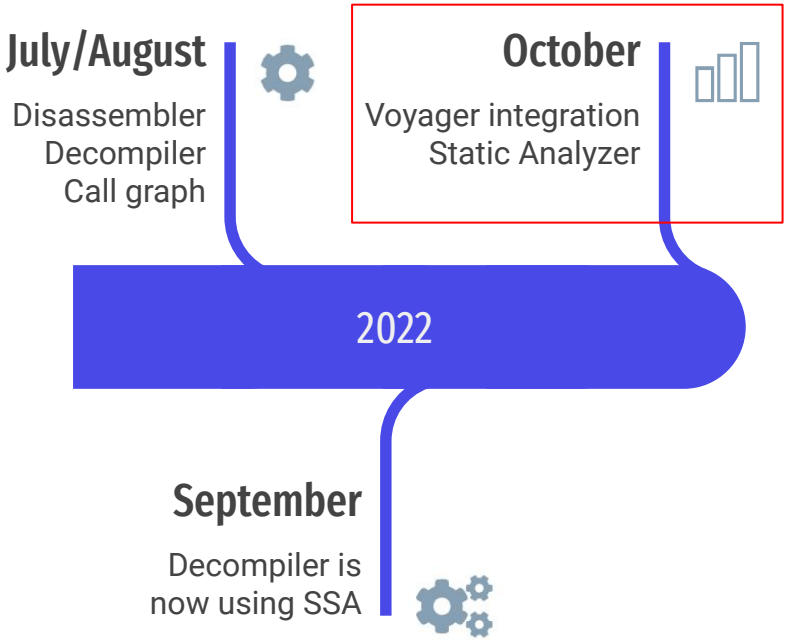
- Thoth 0.1.0

```
func __main__.main{}()
    # 2 -> 0x2
    [AP] = 2;    ap ++
    # 1 -> 0x1
    [AP] = 1;    ap ++
    # 3 -> 0x3
    [AP] = 3;    ap ++
    myfunc([ap-3], [ap-2], [ap-1])
    # 3 -> 0x3
    [AP] = 3;    ap ++
    # 1 -> 0x1
    [AP] = 1;    ap ++
    myfuncbis([ap-2], [ap-1])
    ret
end
```

- Thoth 0.3.0

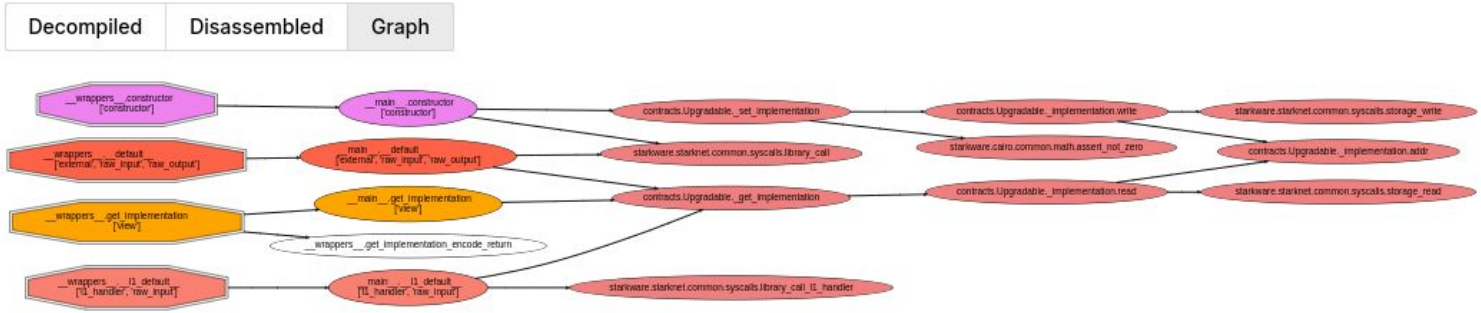
```
// Function 2
func __main__.main{}(){
    v0 = 2    // 0x2
    v1 = 1    // 0x1
    v2 = 3    // 0x3
    myfunc(v0, v1, v2)
    v3 = 3    // 0x3
    v4 = 1    // 0x1
    myfuncbis(v3, v4)
    ret
}
```

Thoth - Evolution



Thoth 0.4.0 - integration inside Voyager

Attribution: This uses Thoth, the Cairo/Starknet bytecode analyzer, disassembler and decompiler created and maintained by FuzzingLabs.



Attribution: This uses Thoth, the Cairo/Starknet bytecode analyzer, disassembler and decompiler created and maintained by FuzzingLabs.

Thoth 0.4.0 - Static Analyzer

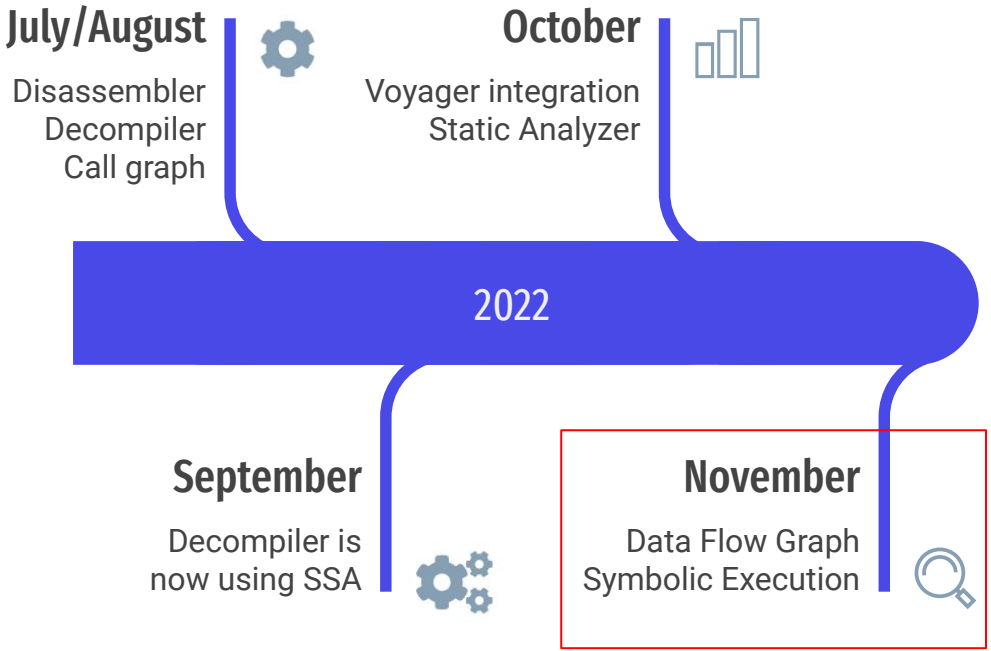
- The analyzer allows to **detect and analyze** particular behaviors in smart contracts.
  - Using the previously extracted information.
- **Analytics**
  - Interesting facts about the contract.
  - ERC detections, strings, etc.
- **Optimization**
  - Detection of potential bytecode optimization.
  - Constants propagation, unused assignment, unused imports, etc.
- **Security**
  - Detection of security vulnerabilities & flaws.
  - Integer overflow, Reentrancy, etc.

Analyzer	Command-Line argument	Description	Impact	Precision	Category
ERC20	`erc20`	Detect if a contract is an ERC20 Token	Informational	High	Analytics
ERC721	`erc721`	Detect if a contract is an ERC721 Token	Informational	High	Analytics
Strings	`strings`	Detect strings inside a contract	Informational	High	Analytics
Functions	`functions`	Retrieve informations about the contract's functions	Informational	High	Analytics
Statistics	`statistics`	General statistics about the contract	Informational	High	Analytics
Assignations	`assignations`	List of variables assignations	Informational	High	Optimization
Integer overflow	`int_overflow`	Detect direct integer overflow/underflow	High	Medium	Security
Function naming	`function_naming`	Detect functions names that are not in snake case	Informational	High	Security
Variable naming	`variable_naming`	Detect variables names that are not in snake case	Informational	High	Security

Thoth 0.4.0 - Example

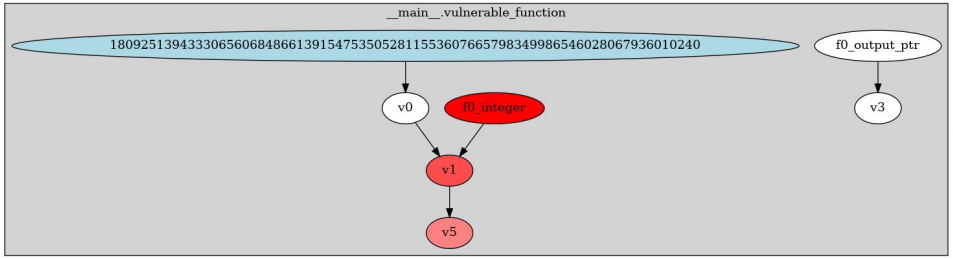
```
~/Documents/thoth/thoth (master) » thoth local ../tests/json_files/cairo_integer_overflow.json -a --color
[Analytics] Functions
  -  (0) vulnerable_function
        - cyclomatic complexity : 1
        - instructions : 6
  -  (1) main (entry point)
        - cyclomatic complexity : 1
        - instructions : 4

[Analytics] Statistics
  -  functions : 3
  -  builtins : 1
  -  structs : 6
  -  calls : 2

[Optimization] Assignations
  -  v0 = 1809251394333065606848661391547535052811553607665798349986546028067936010240
  -  v1 = v0 * f0_integer
  -  v3 = f0_output_ptr
  -  v5 = v1
  -  v6 = f1_output_ptr
  -  v8 = 1
```

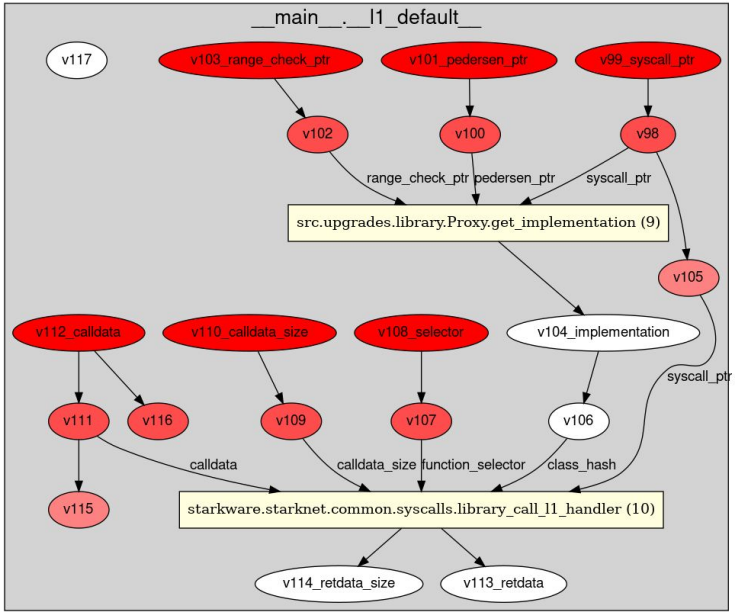
Thoth - Evolution



Data Flow Graph & Tainting

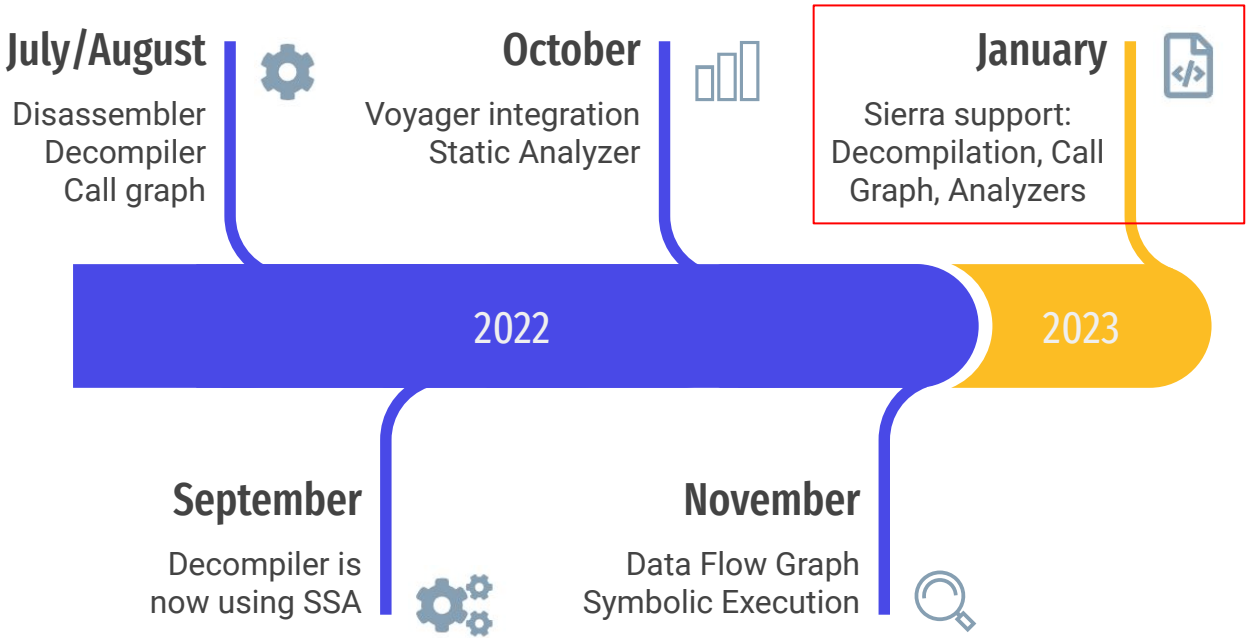
- **Data Flow Graph (DFG)**
  - Variables and constants dependencies representation.
- **Tainting**
  - Allows identifying supplied arguments propagation impact.





- Implement **Symbolic execution**
  - To mathematically solve the constraints to reach certain paths and detect potential issue and/or optimizations.

Thoth - Evolution

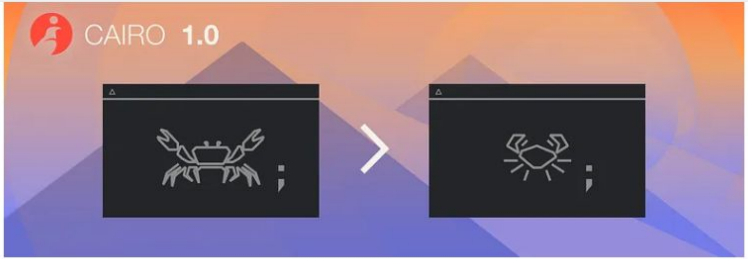


Cairo 1.0 and Sierra

- **Cairo 1.0**
  - New compiler
  - Syntax changed (subset to Rust)
  - More intuitive libraries
  - Improved **expressibility, security, and syntax**.

- Sierra (Safe Intermediate Representation)
  - **Intermediate representation** layer
    - between Cairo 1.0 and Cairo byte code.
  - Ensure that every Cairo run
    - i.e. a Cairo program and its input can be proven

- Further readings:
  - Cairo 1.0 Annoncement - link
  - **Cairo 1.0 is Here - link**
  - Getting Started With Cairo 1.0 - link
  - Cairo 1.0 and Sierra - link
  - Under the hood of Cairo 1.0: Exploring Sierra - link



Cairo 1.0 is Here

Or, as the ancient Egyptians would say, 'Hieroglyphics just got a whole lot easier'

StarkWare · Follow
Published in StarkWare · 3 min read · Jan 5

126 1

TL;DR

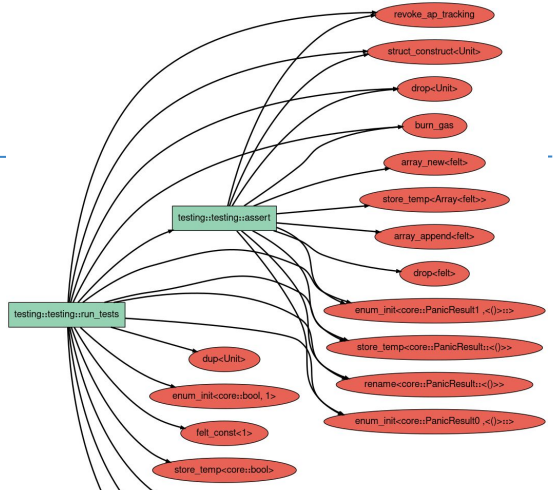
- Cairo 1.0 first release is here!
- Developers can start writing and testing Cairo 1.0 programs
- Feature parity with the older version of Cairo will be reached in the coming weeks
- Support for StarkNet contracts will be added in the upcoming StarkNet Alpha version

Thoth 0.7.0 - Support of Sierra

- Decompilation - link

```
// Function 1
func fib::fib::fib (var_0: felt, var_1: felt, var_2: felt) -> ()
{
        revoke_ap_tracking()
        var_2, var_13 = dup<felt>(var_2)
        if (felt_jump_nz([13]) == 0) {
                drop<NonZero<felt>>(var_3)
                var_1, var_14 = dup<felt>(var_1)
                var_5 = felt_add(var_0, var_14)
                var_6 = felt_const<1>()
                var_7 = felt_sub(var_2, var_6)
                var_9 = store_temp<felt>(var_1)
                var_5 = store_temp<felt>(var_5)
                var_10 = rename<felt>(var_5)
                var_7 = store_temp<felt>(var_7)
                var_11 = rename<felt>(var_7)
                var_8 = function_call<user@fib::fib::fib>(var_9, var_10, var_11)
                var_4 = rename<felt>(var_8)
                burn_gas()
        } else {
                drop<felt>(var_1)
                drop<felt>(var_2)
                var_4 = store_temp<felt>(var_0)
                burn_gas()
        }
        var_12 = rename<felt>(var_4)
        burn_gas()
        return (var_12)
}
```

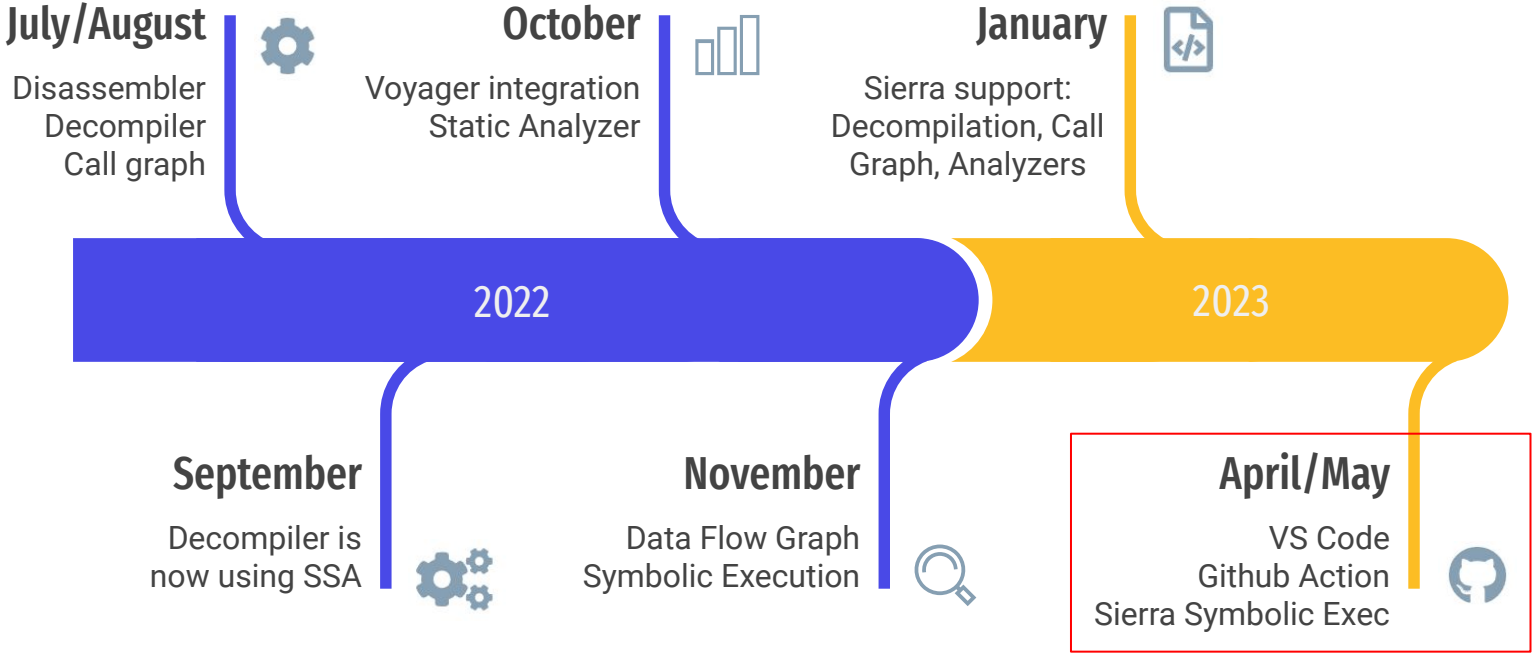
- Call Graph



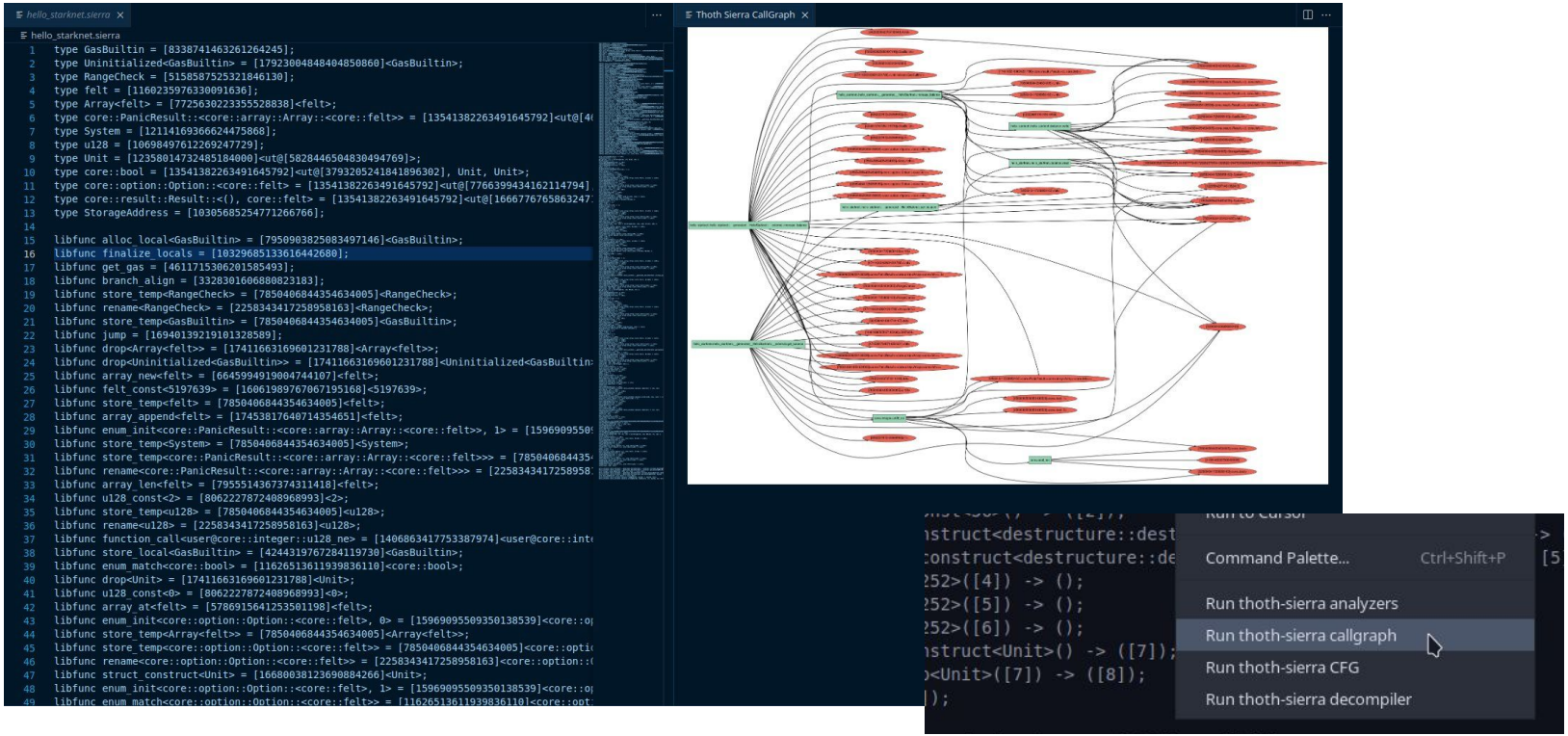
- Analyzer

```
[Analytics] Functions
  -  func cairo_if_list::cairo_if_list::call_if_list (var_0: RangeCheck) -> (Unit)
  -  func core::felt_gt (var_0: RangeCheck, var_1: felt, var_2: felt) -> (core::bool)
  -  func core::felt_lt (var_0: RangeCheck, var_1: felt, var_2: felt) -> (core::bool)
  -  func core::felt_ge (var_0: RangeCheck, var_1: felt, var_2: felt) -> (core::bool)
  -  func core::felt_le (var_0: RangeCheck, var_1: felt, var_2: felt) -> (core::bool)
  -  func core::felt_ne (var_0: RangeCheck, var_1: felt) -> ()
  -  func core::integer::u256_from_felt (var_0: RangeCheck, var_1: felt) -> (core::integer::u256)
  -  func core::integer::u256_lt (var_0: RangeCheck, var_1: felt, var_2: felt) -> (core::bool)
  -  func core::bool_not (var_0: RangeCheck) -> ()
  -  func core::felt_eq (var_0: RangeCheck, var_1: felt) -> ()

[Analytics] Statistics
  -  functions : 10
  -  libfuncs : 49
  -  types : 9

[+] 2 analyzers were run (2 detected)
```

Thoth - Evolution



Thoth 0.8.0 - VS Code plugin & Github action



Thoth 0.8.0 - Sierra Symbolic Execution

- Mathematically **solve the constraints** to reach certain paths/states in the code.
  - Documentation - [link]

```
// Function 1
func symbolic::symbolic::symbolic_execution_test (v0: felt252, v1: felt252, v2: felt252, v3: felt252) -> ()
{
        v4 = felt252_const<102>()
        v5 = felt252_sub(v0, v4)
        v5 = store_temp<felt252>(v5)
        if (felt252_is_zero(v5) == 0) {
                branch_align()
                drop<NonZero<felt252>>(v6)
        } else {
                branch_align()
        }
        v7 = felt252_const<117>()
        v8 = felt252_sub(v1, v7)
        v8 = store_temp<felt252>(v8)
        if (felt252_is_zero(v8) == 0) {
                branch_align()
                drop<NonZero<felt252>>(v9)
        } else {
                branch_align()
        }
        v10 = felt252_const<122>()
        v11 = felt252_sub(v2, v10)
        v11 = store_temp<felt252>(v11)
        if (felt252_is_zero(v11) == 0) {
                branch_align()
                drop<NonZero<felt252>>(v12)
        } else {
                branch_align()
        }
```

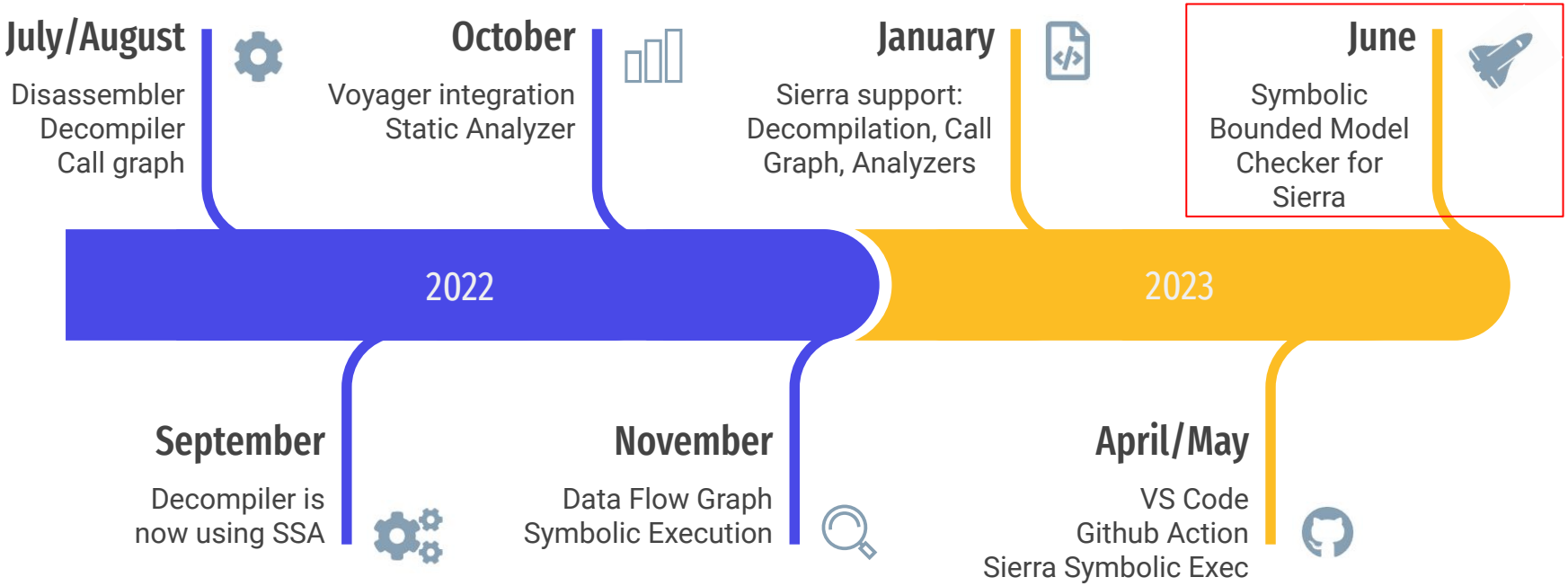
Variable to solve
- v0 v1 v2 v3

Constraints
- v5==0
- v8==0
- v11==0
- v14==0

Z3

```
v0: 102
v1: 117
v2: 122
v3: 122
```

Thoth - Evolution



Thoth 0.9.0 - Symbolic Bounded Model Checker for Sierra

- Thoth-checker - Symbolic Bounded Model Checker for Sierra
  - **Formal verification** of testing function written directly in Cairo
  - Equivalent in EVM: a16z/halmos

- Check failed

```
fn thoth_test_product(mut a: felt252) {
    let c = a * 2;
    assert(c == 11, '');
}
```

```
$ ~ thoth-checker -f ./test_checker_2.sierra

[+] Thoth Symbolic bounded model checker

[FAIL] test_checker::test_checker::thoth_test_product
```

- Check succeeded

```
fn add(mut a: felt252, mut b: felt 252) -> felt252 {
    let c = a + b;
    c
}

fn thoth_test_sum() {
    let sum = add(1, 2);
    assert(sum == 3, '');
}
```

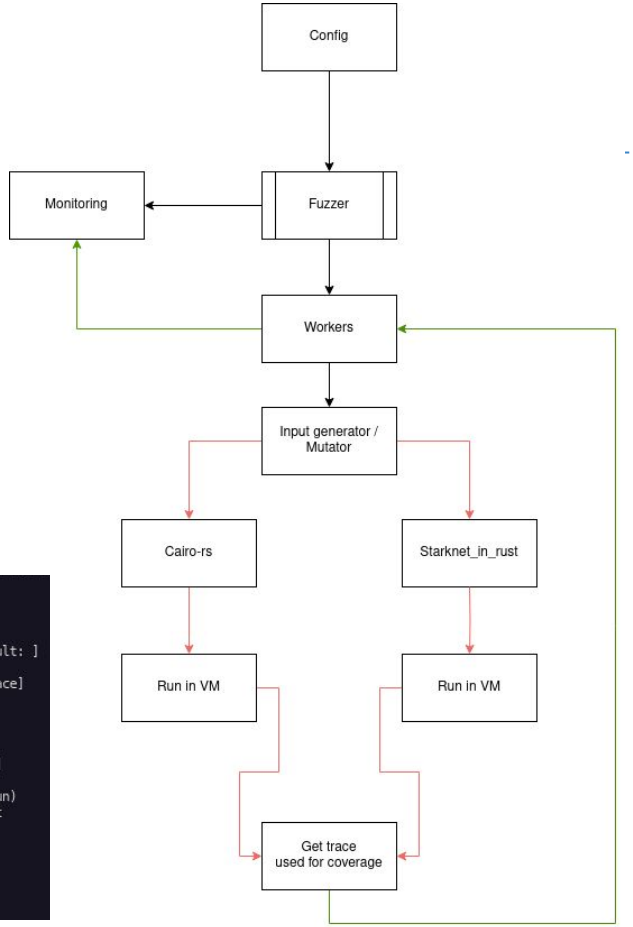
```
$ ~ thoth-checker -f ./test_checker_3.sierra

[+] Thoth Symbolic bounded model checker

test_checker::test_checker::thoth_test_sum SUCCESS
```

Cairo-fuzzer

Cairo-Fuzzer - Architecture

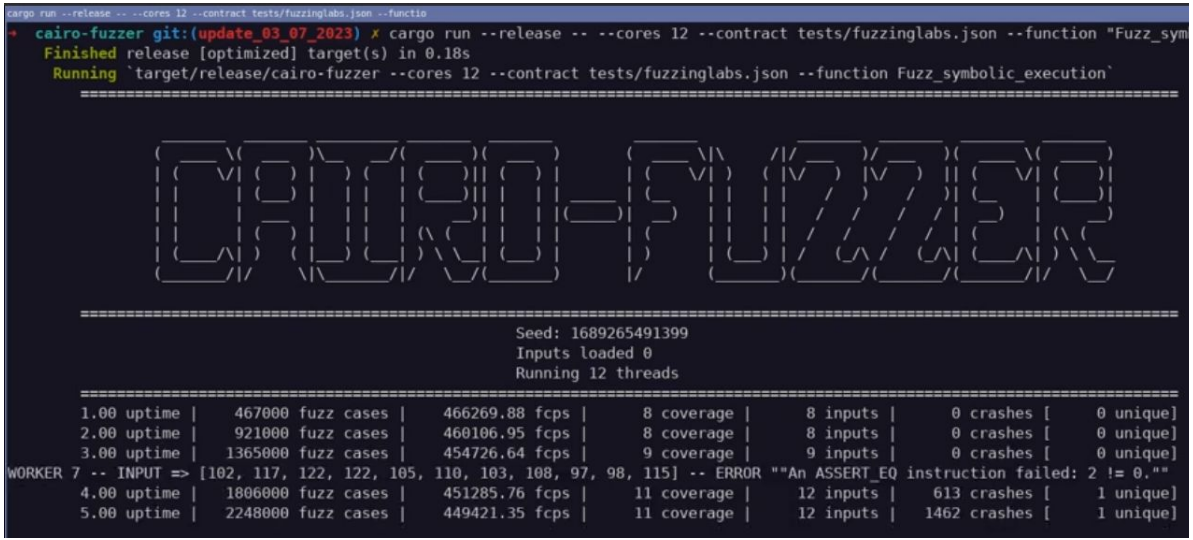
- Architecture
  - Coverage-guided
  - Multithreaded with good scalability
    - 70k exec/s for 1 thread
    - 440k exec/s for 10 threads
  - Execution engines
    - lambdaclass/cairo-vm for Cairo contract
    - lambdaclass/starknet_in_rust for StarkNet contract
  - Usable as a library
- Features
  - Property testing
  - Minimizer
  - Replayer
  - Usage of dictionary
  - etc.

```
Usage: cairo-fuzzer [OPTIONS]

Options:
      --cores <CORES>              Set the number of threads to run [default: 1]
      --contract <CONTRACT>        Set the path of the JSON artifact to load [default: ]
      --function <FUNCTION>        Set the function to fuzz [default: ]
      --workspace <WORKSPACE>      Workspace of the fuzzer [default: fuzzer_workspace]
      --inputfolder <INPUTFOLDER>  Path to the inputs folder to load [default: ]
      --crashfolder <CRASHFOLDER>  Path to the crashes folder to load [default: ]
      --inputfile <INPUTFILE>      Path to the inputs file to load [default: ]
      --crashfile <CRASHFILE>      Path to the crashes file to load [default: ]
      --dict <DICT>                Path to the dictionnary file to load [default: ]
      --logs                       Enable fuzzer logs in file
      --seed <SEED>                Set a custom seed (only applicable for 1 core run)
      --run-time <RUN_TIME>        Number of seconds this fuzzing session will last
      --config <CONFIG>            Load config file
      --replay                     Replay the corpus folder
      --minimizer                  Minimize Corpora
      --proptesting                Property Testing
      --iter <ITER>                Iteration Number [default: -1]
  -h, --help                       Print help
```



Cairo-Fuzzer - Example

- 12 cores
- 460k exec/seconds
- [demo]




```
func Fuzz_symbolic_execution(
    f: felt,
    u: felt,
    z: felt,
    z2: felt,
    i: felt,
    n: felt,
    g: felt,
    l: felt,
    a: felt,
    b: felt,
    s: felt,
) {
    if (f == 'f') {
        if (u == 'u') {
            if (z == 'z') {
                if (z2 == 'z') {
                    if (i == 'i') {
                        if (n == 'n') {
                            if (g == 'g') {
                                if (l == 'l') {
                                    if (a == 'a') {
                                        if (b == 'b') {
                                            if (s == 's') {
                                                assert 0 = 2;
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    return ();
}
```

Future developments

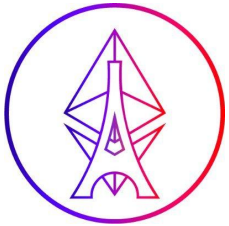
Future developments

- Future of thoth & cairo-fuzzer
  - Speed and documentation improvements
  - Tutorials

- Planning to build more fuzzer & security tools!
  - WebAssembly
  - Substrate
  - Cosmos/CosmWasm
  - Algorand
  - etc.

- **Contacts us** if you need **customs security tool** development!
  - Twitter: @Pat_Ventuzelo
  - Mail: patrick@fuzzinglabs.com

- Thanks for your time! Any questions?



FUZZING LABS