Unlocking Secret Analysis in GCC Static Analyzer

Pierrick Philippe

Univ. Ren., CNRS, IRISA, France
PhD student, SPICY team
Director and advisor: Pierre-Alain Fouque, Mohamed Sabt

FOSDEM'24

February 4 2024

Université de Rennes

DGA DIRECTION GÉNÉRALE DE L'ARMEMENT

UMR IRISA

SPICY

GCC Static Analyzer Example

```
1  int main(void) {
2    int *ptr = (int *) malloc(sizeof(int));
3
4    if (ptr) {
5      free(ptr);
6      *ptr = 42;
7    }
8
9    return 0;
10 }
```

GCC Static Analyzer Example

```
gcc -fanalyzer test.c
```

```
./malloc.c: In function 'main':
./malloc.c:8:10: warning: use after 'free' of 'ptr' [CWE-416] [-Wanalyzer-use-after-free]
    8 |     *ptr = 42;
      |     ~~~~~^~~~
  'main': events 1-6
    |
    |    4 |   int * ptr = (int *) malloc(sizeof(int));
    |      |                       ^~~~~~~~~~~~~~~~~~~
    |      |                       |
    |      |                       (1) allocated here
    |    5 |
    |    6 |   if (ptr) {
    |      |      ~
    |      |      |
    |      |      (2) assuming 'ptr' is non-NULL
    |      |      (3) following 'true' branch (when 'ptr' is non-NULL)...
    |    7 |     free(ptr);
    |      |     ~~~~~~~~~
    |      |     |
    |      |     (4) ...to here
    |      |     (5) freed here
    |    8 |     *ptr = 42;
    |      |     ~~~~~~~~~
    |      |          |
    |      |          (6) use after 'free' of 'ptr'; freed at (5)
    |
```

GCC Static Analyzer Overview

- introduced in GCC 10
- API available for out-of-tree plugin
- symbolic execution

GCC Static Analyzer Overview

- introduced in GCC 10
- API available for out-of-tree plugin
- symbolic execution $\rightarrow$ path feasability

GCC Static Analyzer Overview

- introduced in GCC 10
- API available for out-of-tree plugin
- symbolic execution $\rightarrow$ path feasability
- state machine internally managed
- reporting system:
  - really nice formatted output on stdout
  - SARIF file

GCC Static Analyzer Overview

- introduced in GCC 10
- API available for out-of-tree plugin
- symbolic execution → path feasability
- state machine internally managed
- reporting system:
  - really nice formatted output on stdout
  - SARIF file

Thanks to David Malcolm for his work on the SA[1]!

[1] And also answering my GCC noobie emails 😇

GCC Static Analyzer Internals Overview

Variables `lvalue` and `rvalue` representation:

```
1 int x = 42;
2 int *ptr = &x;
```

ana::store	
ana::region (`lvalue`)	*ana::svalue* (`rvalue`)
x	42
ptr	&x

GCC Static Analyzer State Machine 101

Implementing a state machine:

- inherit from *ana::state_machine*

GCC Static Analyzer State Machine 101

Implementing a state machine:

- inherit from *ana::state_machine*
- SA handles a map of *svalue* to *state* for you
  - can have another *svalue* as origin for reporting

GCC Static Analyzer State Machine 101

Implementing a state machine:

- inherit from *ana::state_machine*
- SA handles a map of *svalue* to *state* for you
  - can have another *svalue* as origin for reporting
- states are tunable:
  - inherit from *ana::state*

GCC Static Analyzer State Machine 101

Implementing a state machine:

- inherit from *ana::state_machine*
- SA handles a map of *svalue* to *state* for you
  - can have another *svalue* as origin for reporting
- states are tunable:
  - inherit from *ana::state*

And that's all you need to start!

Taint Analysis Overview

User input validation: SQL, OS kernel, etc.

Taint Analysis Overview

User input validation: SQL, OS kernel, etc.

Core ideas:

- source
- propagator
- sink
- filter

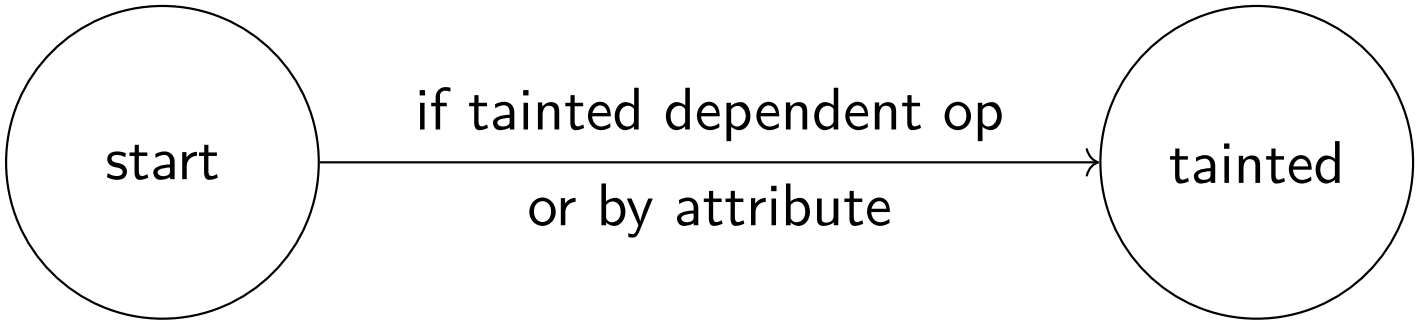
Taint Analysis Applied to Secret Analysis

User input validation: SQL, OS kernel, etc.

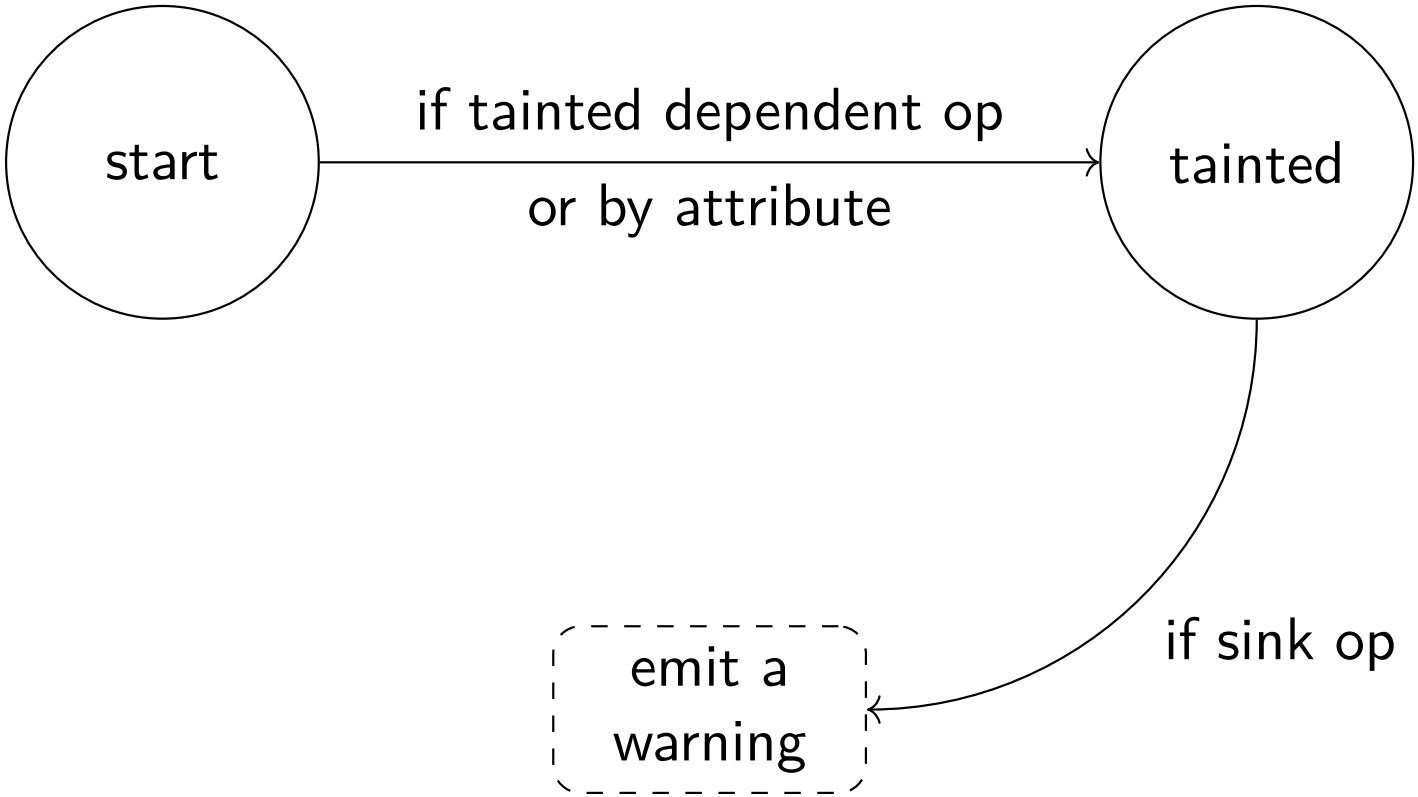
Core ideas:

- source → secret
- propagator → taint propagation implementation
- sink → condition, memory access, non-constant CPU operations
- filter → taint-erasure (e.g., `bzero`)

State Machine Secret Analysis



State Machine Secret Analysis



State Machine Example 1

```
1  int TAINT_ATTR secret = 42;
2  // secret now tracked as tainted
3
4  if (secret) // emitting a warning
5    doSomething();
```

State Machine Example 2

```
1  int TAINT_ATTR secret = 42;
2  // secret now tracked as tainted
3
4  int y = secret + 100;
5  // y now tracked as tainted
6
7  if (y) // emitting a warning
8    doSomething();
```

State Machine Example 2

ana::store	
Key (*region*)	Value (*svalue*)
secret	42
y	142

ana::sm_state_map	
Key (*svalue*)	Value (*state*)
42	tainted
142	tainted

State Machine Problematic Example

```
1  int TAINT_ATTR secret = 42;
2  // secret now tracked as tainted
3
4  int y = 42;
5  // y implictly tracked as tainted
6
7  if (y) // emitting a false warning
8    doSomething();
```

State Machine Problematic Example

ana::store	
Key (*region*)	Value (*svalue*)
secret	42
y	42

ana::sm_state_map	
Key (*svalue*)	Value (*state*)
42	tainted

Enhancing *ana::sm_state_map*

ana::store	
Key (*region*)	Value (*svalue*)
secret	42
y	42

ana::sm_state_map	
Key (*svalue*)	Value (*state*)
empty	
Key (*region*)	Value (*state*)
secret	tainted

Enhancing *ana::sm_state_map*

```
1 int TAINT_ATTR secret = 42;
2 // secret now tracked as tainted
3
4 int y = 42;
5 // y not implictly tracked anymore :)
6
7 if (y) // no warning emitting
8   doSomething();
```

Enhancing *ana::sm_state_map*

```
1  int TAINT_ATTR secret = 42;
2  // secret now tracked as tainted
3
4  int y = secret + 100;
5  // y now tracked as tainted
6
7  int *ptr = &y;
8  // ptr now tracked as tainted
9
10 if (*ptr) // emitting a warning
11   doSomething();
```

Enhancing *ana::sm_state_map*

ana::sm_state_map		
Key (*svalue*)	Value (*entry_t*)	
	state	*origin_t*
&y	tainted	(*region*) y
Key (*region*)	Value (*entry_t*)	
	state	*origin_t*
secret	tainted	nullptr
y	tainted	(*region*) secret

SA Modifications Overview

- *ana::sm_state_map*
  - able to track *region*'s state
  - origin can be either *svalue* or *region*
  - modified/added some API
- *ana::sm_context*
  - modified/added some API
- diagnostic related code

Scalar Array/Pointer Aliasing

```
1  int TAINT_ATTR secret = 42;
2  // secret is tainted
3  int t[4] = { 0 };
4  int *ptr = t + 2;
5
6  t[2] = secret;
7  // t[2] is tainted
8
9  // emitting a warning as ptr alias t + 2
10 if (*ptr)
11   doSomething();
```

Scalar Array/Pointer Aliasing

ana::store	
Key (*region*)	Value (*svalue*)
secret	42
t	[0, 0, 42, 0]
ptr	&t + 8

ana::sm_state_map		
svalue	*entry_t*	
	state	*origin_t*
&t + 8	tainted	secret
region	*entry_t*	
	state	*origin_t*
secret	tainted	nullptr
t[2]	tainted	secret

Interprocedural Analysis

```
1  int main (void) {
2    int TAINT_ATTR secret = 42;
3
4    f(secret);
5
6    return 0;
7  }
8
9  void f (int x) {
10   if (x) // warning should be emitted
11     doSomething();
12 }
```

Interprocedural Analysis

Frame #0 (*main*)		
store		
region	*svalue*	
secret	*42*	
sm_state_map		
region	*state*	*origin*
secret	tainted	nullptr

Frame #1 (*f*)		
store		
region	*svalue*	
x	*42*	
sm_state_map		
region	*state*	*origin*
x	tainted	secret (main)

Takeaways

- possible to track state for *region*
- issues remains with:
  - scalar arrays/pointer aliasing
  - interprocedural analysis

Let's discuss about it! 🍻

Feel free to reach out:
✉ pierrick.philippe@irisa.fr