Reverse Engineering

Metodik och praktiska tips för CTF

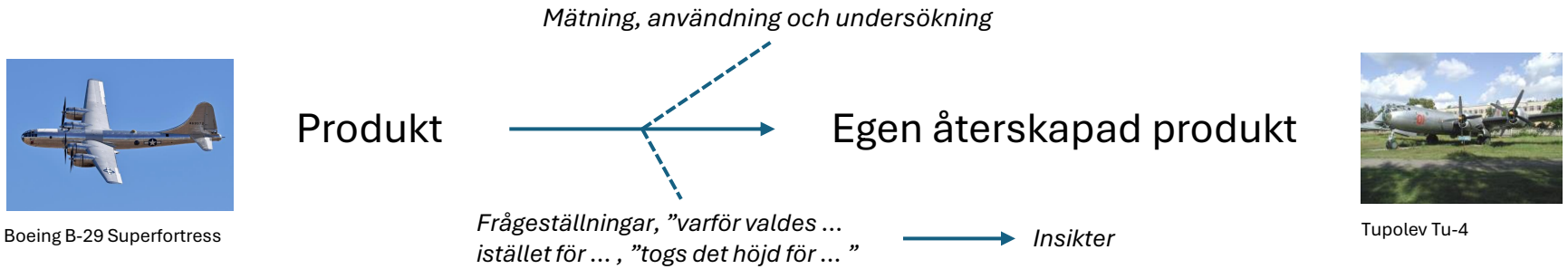
Agenda

- Bakgrund
- Metodik med ett exempel
- Utmaningar

Bakgrund

- Reverse Engineering

  Återskapa förståelse/produkt med minimala ingångsvärden genom användning, undersökning och analys



Boeing B-29 Superfortress

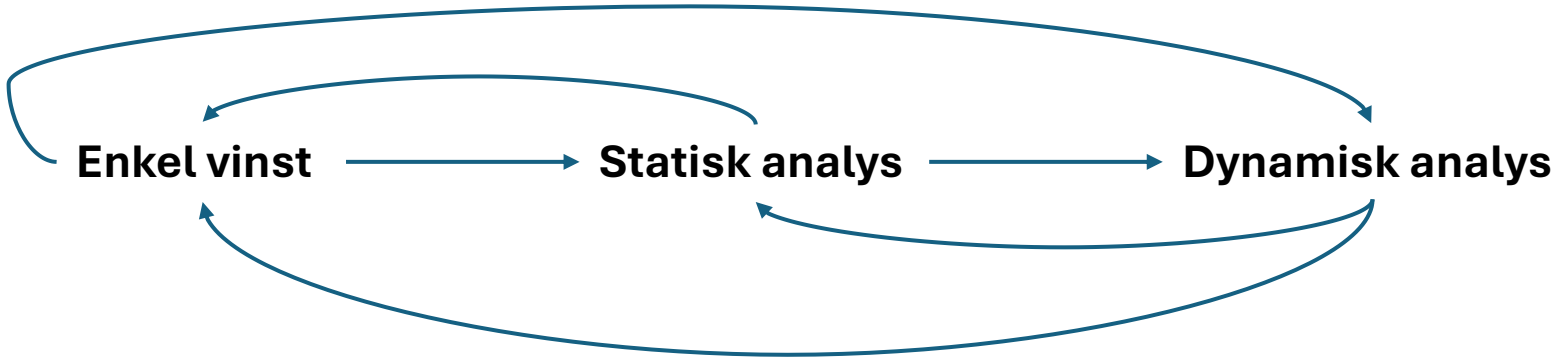
Tupolev Tu-4

- Software Reverse Engineering

  Återskapa förståelse/mjukvara utan ingångsvärden genom användning, undersökning och analys

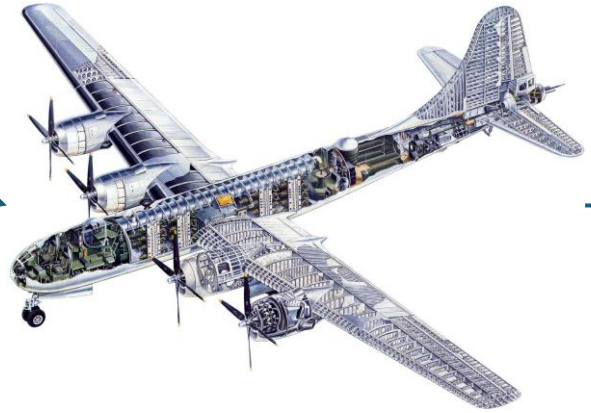
Metodik

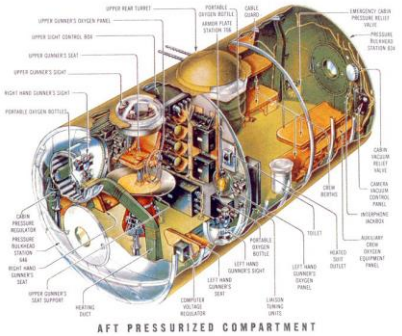
- Reverse Engineering / Software Reverse Engineering
- Enkla vinster / Undersök binär
- Statisk analys
- Dynamisk analys

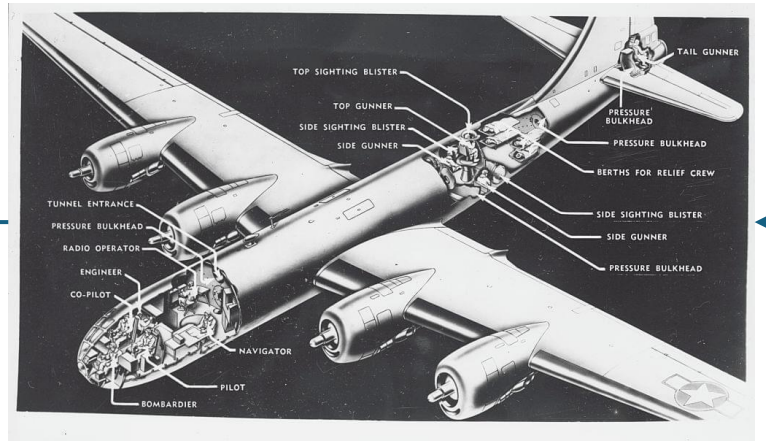


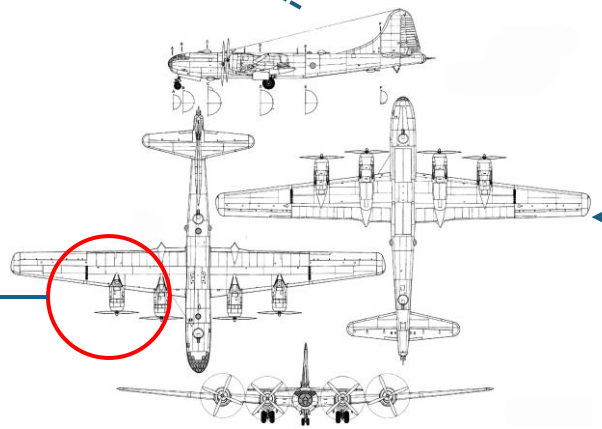
Metodik – Reverse Engineering

Boeing B-29 Superfortress







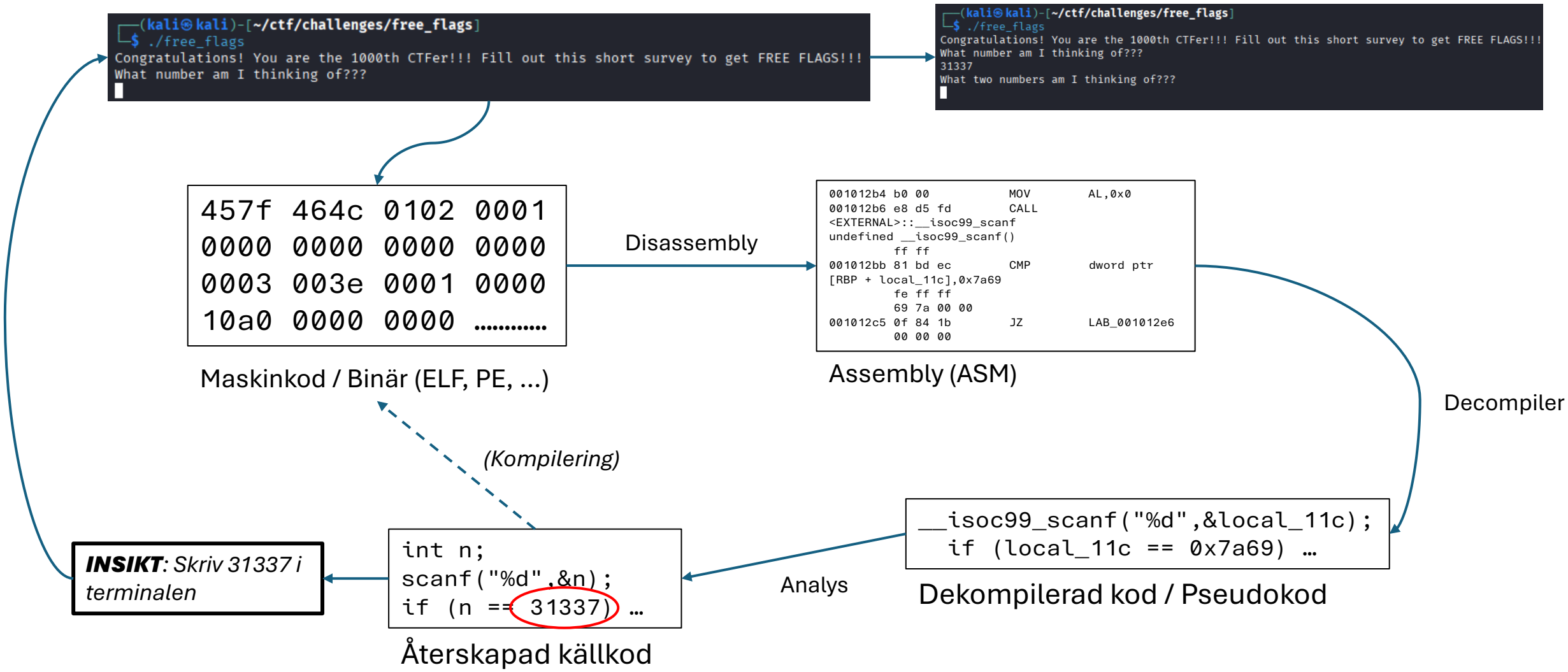


INSIKT: …

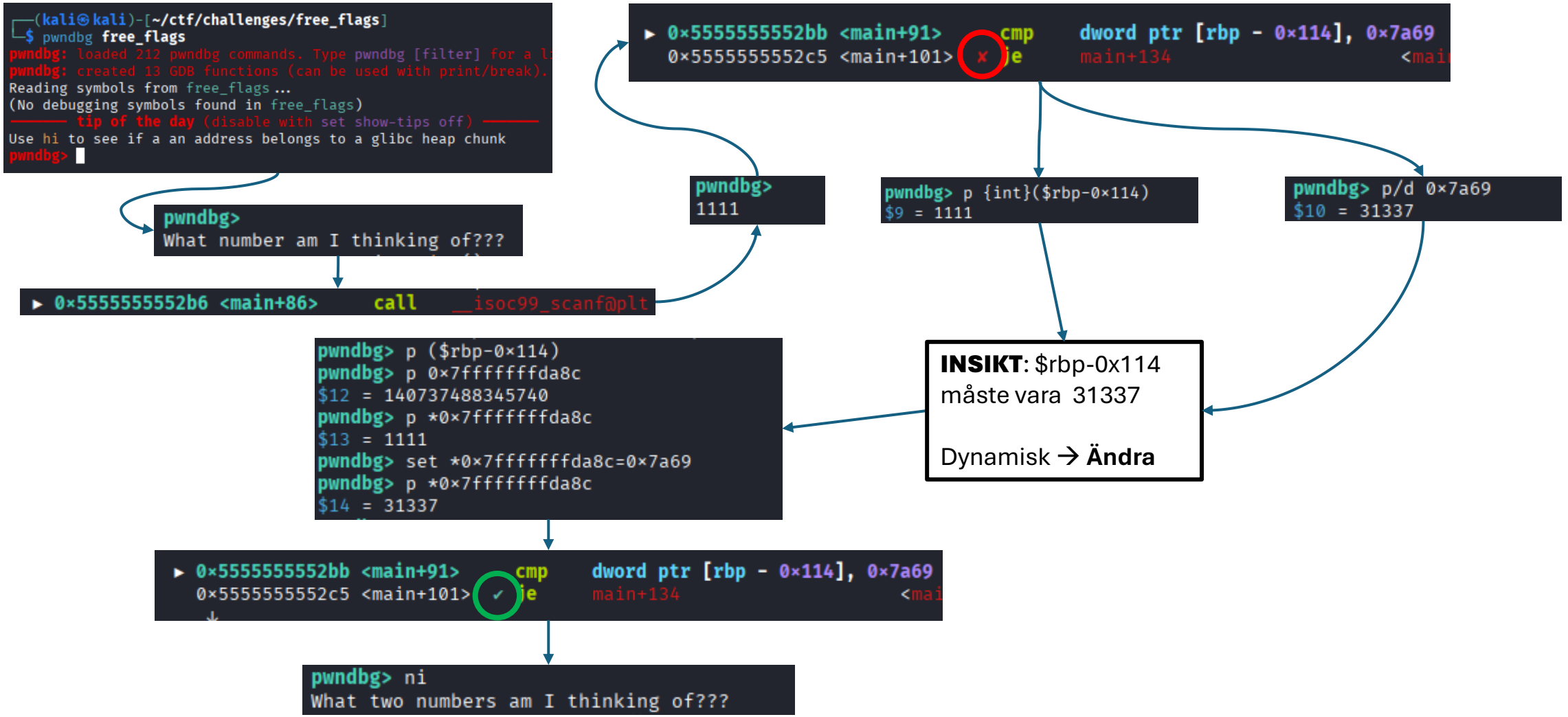
Metodik – Enkla vinster

- FILE: `` `file [filename]` ``
  - Identifiera vilket typ av fil samt andra attributer.
  - Leta efter **PIE** → *Position Independent Execution*, Alla addresser blir relative (RVA) istället för VA, offset istället för absoluta adresser
  - Leta efter **STRIPPED / NOT STRIPPED** → Huruvida om debug-symboler (”main” funktionen) är inkluderad i binären
- STRINGS: `` `strings -n {N} [filename]` ``
  - Printar alla utskrivningsbara tecken i filen
  - Ställ in en rimlig N, **10 är bra**
  - Leta efter intressanta strängar med ex: `` `strings -n 10 [filename] | grep [needle]` ``

Metodik – Statisk analys med Ghidra



Metodik – Dynamisk analys med GBD/PWNDBG



Metodik – Analys med Python

```
puts("What two numbers am I thinking of???");
__isoc99_scanf("%d %d",&num1,&num2);
if ((num1 + num2 == 1142) && (num1 * num2 == 302937)) {
```

INSIKT: Två nummer måste summera till **1142** samt multipliceras till **302937**

```
def factors(n):
    f = []
    for x in range (1,n+1):
        if n%x==0:
            f.append(x)
    return f

if __name__=='__main__':
    print(factors(int(302937)))
```

[1, 3, 241, 419, 723, 1257, 100979, 302937]

419+723=1142

419*723=302937

INSIKT: De två nummer som uppfyller vilkoren är **419** och **723**

```
┌──(kali㉿kali)-[~/ctf/challenges/free_flags]
└─$ ./free_flags
Congratulations! You are the 1000th CTFer!!! Fill out this short survey to get FREE FLAGS!!!
What number am I thinking of???
31337
What two numbers am I thinking of???
419
723
What animal am I thinking of???
```

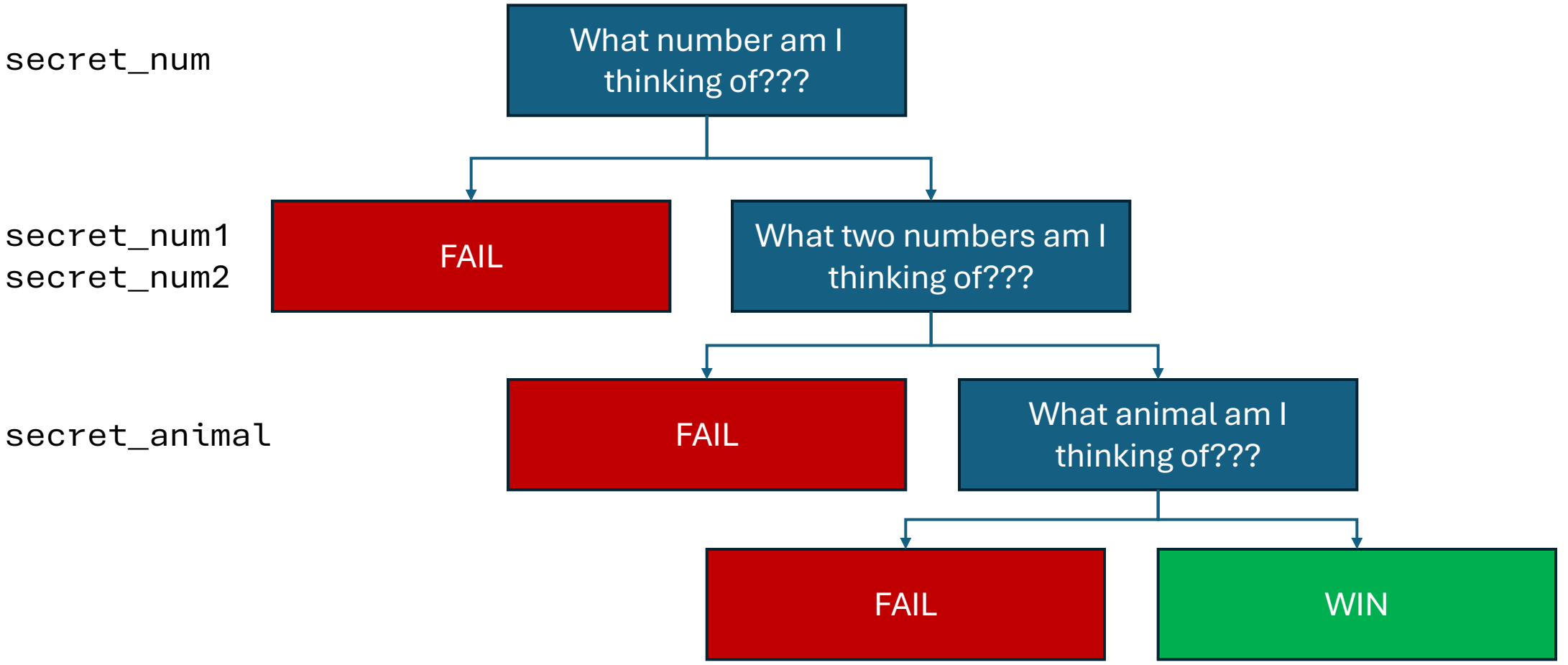
Metodik – Dynamisk analys med LTRACE

```
┌──(kali㉿kali)-[~/ctf/challenges/free_flags]
└─$ ./free_flags
Congratulations! You are the 1000th CTFer!!! Fill out this short survey to get FREE FLAGS!!!
What number am I thinking of???
31337
What two numbers am I thinking of???
419
723
What animal am I thinking of???
```

```
┌──(kali㉿kali)-[~/ctf/challenges/free_flags]
└─$ ltrace ./free_flags
puts("Congratulations! You are the 100" ... Congratulations! You are the 1000th CTFer!!! Fill out this short survey to get FREE FLAGS!!!
)                                                                 = 93
puts("What number am I thinking of???"What number am I thinking of???
)                                                                 = 32
__isoc99_scanf(0×55abf91cc0e1, 0×7ffe1c851a9c, 0, 0
puts("What two numbers am I thinking o" ... What two numbers am I thinking of???
)                                                                 = 37
__isoc99_scanf(0×55abf91cc0de, 0×7ffe1c851a98, 0×7ffe1c851a94, 0×7ffe1c851cd8
puts("What animal am I thinking of???"What animal am I thinking of???
)                                                                 = 32
__isoc99_scanf(0×55abf91cc104, 0×7ffe1c851aa0, 0, 0×7ffe1c851cd8hippo
)                                                 = 1
strcspn("hippo", "\n")
strcmp("hippo", "banana")
puts("Wrong >:((((Wrong >:((((
)
+++ exited (status 1) +++
```

```
┌──(kali㉿kali)-[~/ctf/challenges/free_flags]
└─$ ./free_flags
Congratulations! You are the 1000th CTFer!!! Fill out this short survey to get FREE FLAGS!!!
What number am I thinking of???
31337
What two numbers am I thinking of???
419
723
What animal am I thinking of???
banana
Wow!!! Now I can sell your information to the Russian government!!!
Oh yeah, here's the FREE FLAG:
ctf{free_flags_are_the_best}
```

Metodik – Statisk analys med SMT/Z3



secret_num, secret_num1, secret_num2, `secret_animal` → `WIN/FAIL`

Metodik – Statisk analys med SMT/Z3

CONSTRAINTS/VILLKOR

```
secret_num == 31337

num1+num2 == 1142
num1*num2 == 302937

secret_animal == banana
```

Satisfiability modulo theories (SMT)

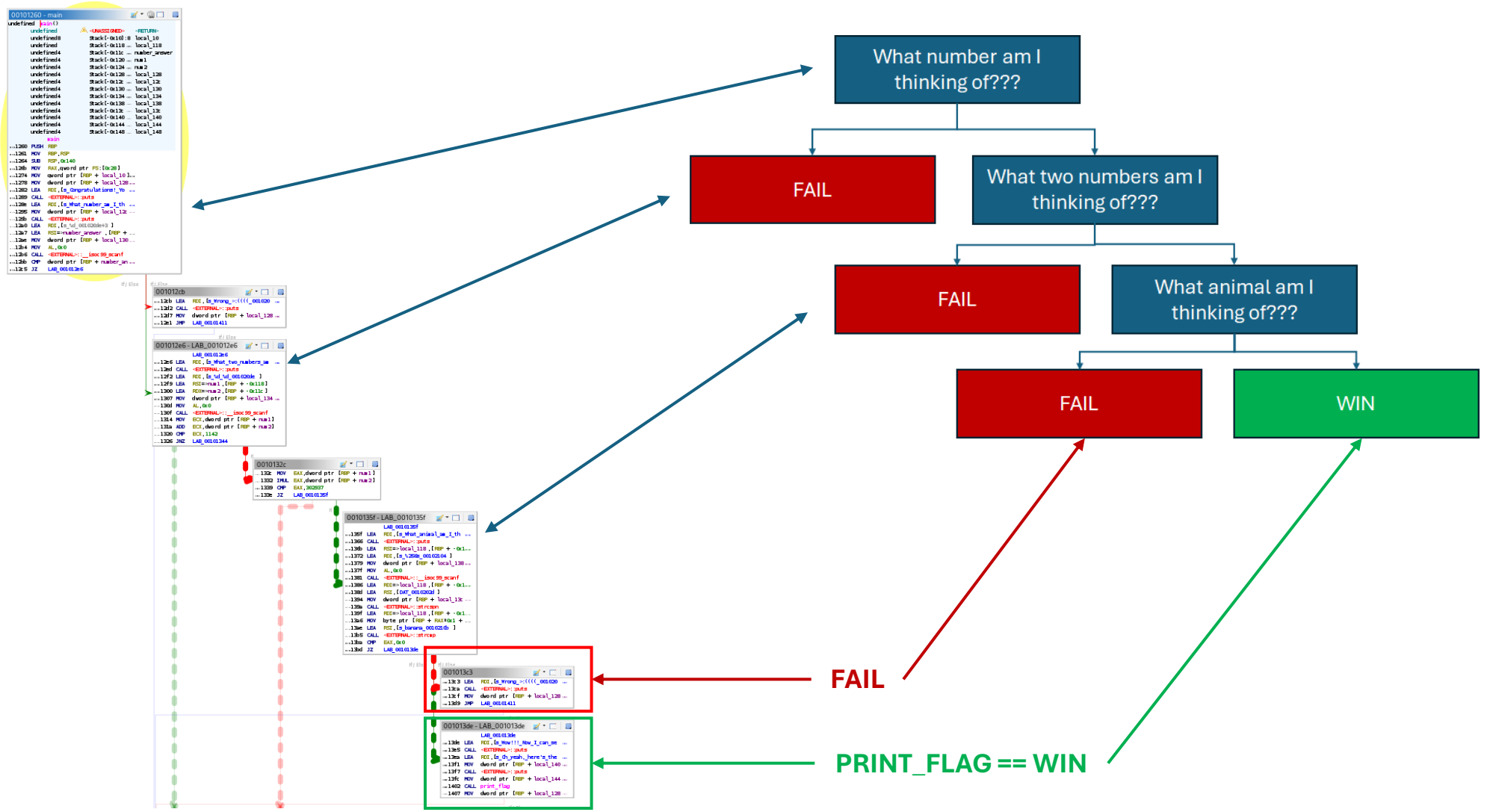
Are the constrainsts satisfiable?

Z3 *– solver library for symbolic logic (eg: x>1 and x< 4)*

```
from z3 import *

solver = Solver()

secret_num = Int('secret_num')
num1 = Int('num1')
num2 = Int('num2')
secret_animal = String('secret_animal')

solver.add(secret_num == 31337)
solver.add(num1 + num2 == 1142)
solver.add(num1 * num2 == 302937)
solver.add(secret_animal == "banana")

if solver.check() == sat:
    model = solver.model()
    print(model)
else:
    print("No solution found.")
```

```
┌──(.venv)─(kali㉿kali)-[~/ctf/challenges/free_flags]
└─$ python3 z3_solution.py
[num2 = 419,
 secret_animal = "banana",
 num1 = 723,
 secret_num = 31337]
```

Metodik – Dynamisk analys med ANGR



Metodik – Dynamisk analys med ANGR

SMT + Symbolic Execution + Call graph + … == ANGR

Is there a path to the desired program state

```python
import angr
project = angr.Project('free_flags', auto_load_libs=False)

find_symbol = "print_flag"
find_addr = project.loader.find_symbol(find_symbol)

initial_state = project.factory.entry_state()
simgr = project.factory.simulation_manager(initial_state)
simgr.explore(find=find_addr.rebased_addr)

if simgr.found:
    print("Found a path to the print_flag function.")
    solution_state = simgr.found[0]
    solution_input = solution_state.posix.dumps(0)
    print("Solution:", solution_input.decode("utf-8"))
else:
    print("No solution found!")
```

```
┌──(.venv)─(kali㉿kali)-[~/ctf/challenges/free_flags]
└─$ python3 angr_solution.py
Found 'print_flag' function at address: 0×4011a0
Found a path to the print_flag function.
Solution: 00000313370000000419 0000000723 banana
```

0000031337 0000000419 0000000723 banana

31337
419
723
banana

Metodik – Sammanfattning

- Enkla vinster / Undersök binären → file, strings
- Statisk analys → Ghidra, Z3, Python
- Dynamisk analys → GDB/pwndbg, ANGR, LTRACE

Resurser

- Ghidra:
  - https://ghidra-sre.org/
  - https://ghidra.re/
- Z3:
  - https://ericpony.github.io/z3py-tutorial
  - https://pypi.org/project/z3-solver/
- GDB/pwndbg:
  - https://pwndbg.re/stable/
- ANGR
  - https://angr.io/

Utmaningar

- strings → https://chall.472.sh/strings
- You cant C me → https://chall.472.sh/you_cant_c_me
- check → https://chall.472.sh/check
- free_flags → https://chall.472.sh/free_flags
- access → https://chall.472.sh/access
- helithumper → https://chall.472.sh/helithumper
- anti_flag → https://chall.472.sh/anti_flag
- jailbreak → https://chall.472.sh/jailbreak
- ircware → https://chall.472.sh/ircware