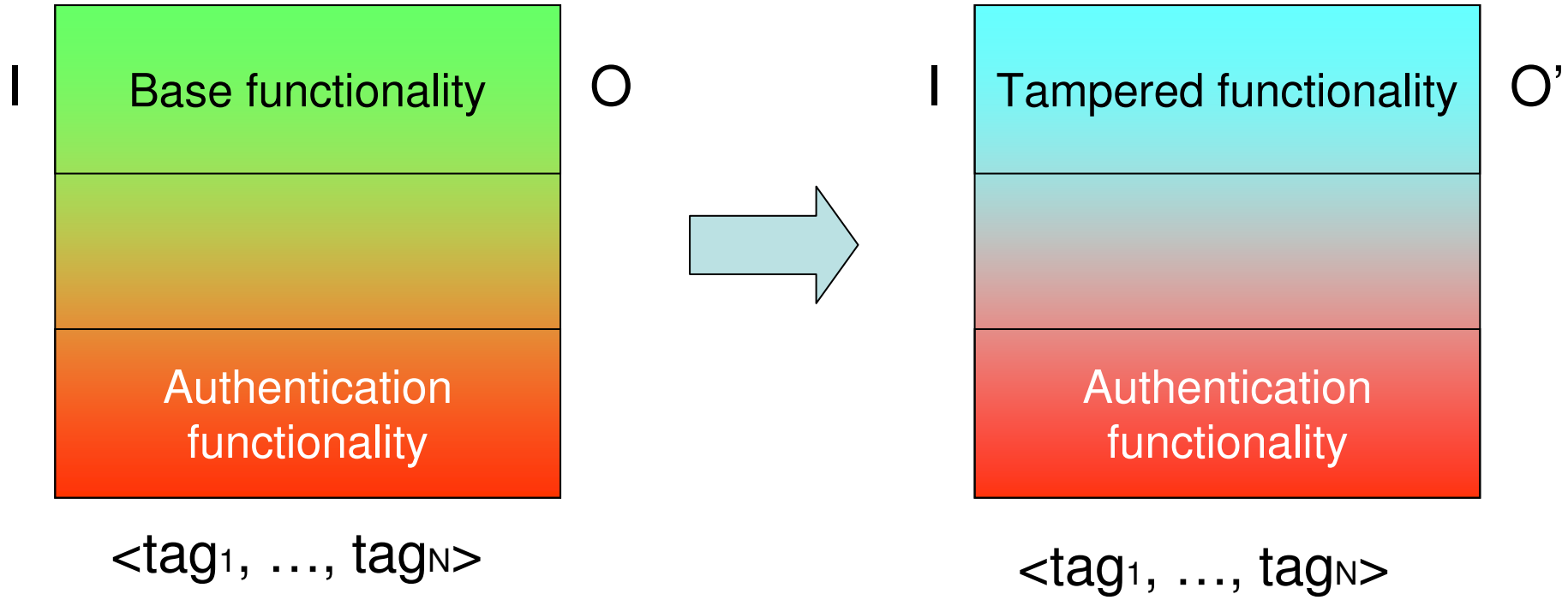
Reverse engineering and mutation analysis

in the context of software tampering and authentication

Paolo Tonella
ITC-irst, Centro per la Ricerca Scientifica e Tecnologica
Povo, Trento, Italy
tonella@itc.it

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

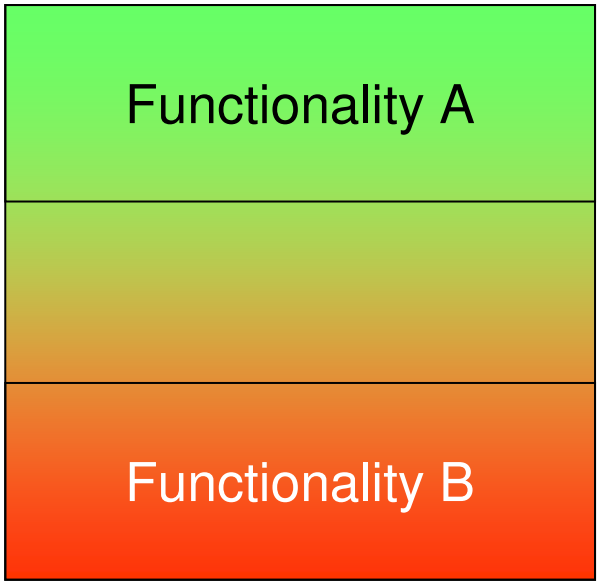
Software tampering



What are the basic problems underlying software tampering?

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

Reverse engineering

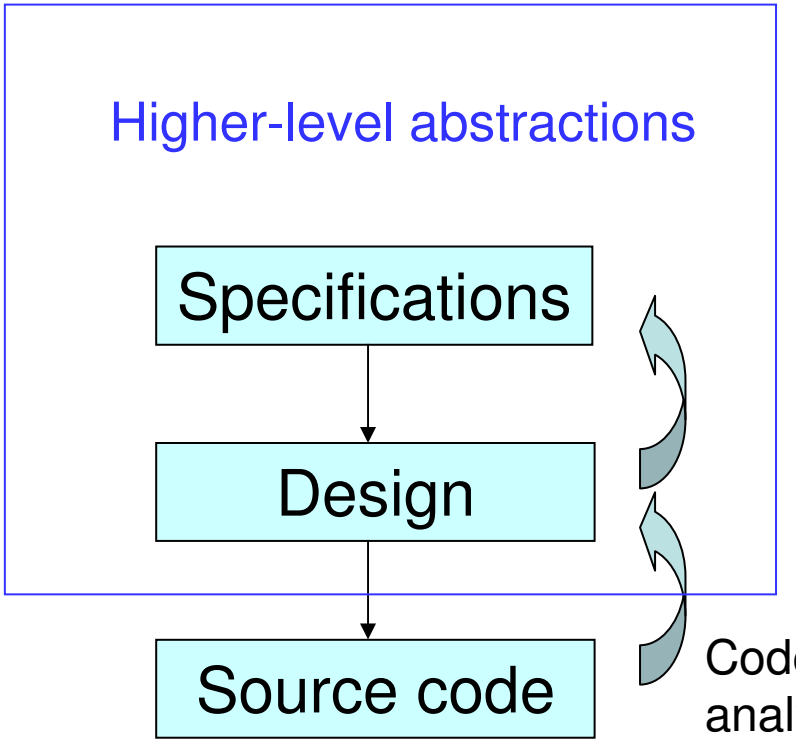


- **Feature location**: where is func. A (B) implemented?
- **Slicing**: what subprogram computes func. A (B)?
- **Impact analysis**: what are the ripple effects of changing given statements of func A?

These problems have been deeply investigated in the area of reverse engineering.

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

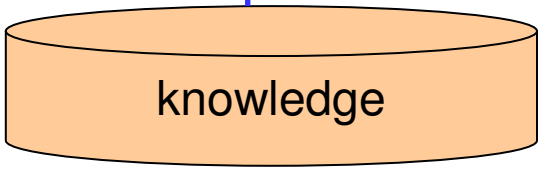
Context of reverse engineering



Code analysis

Goal: supporting

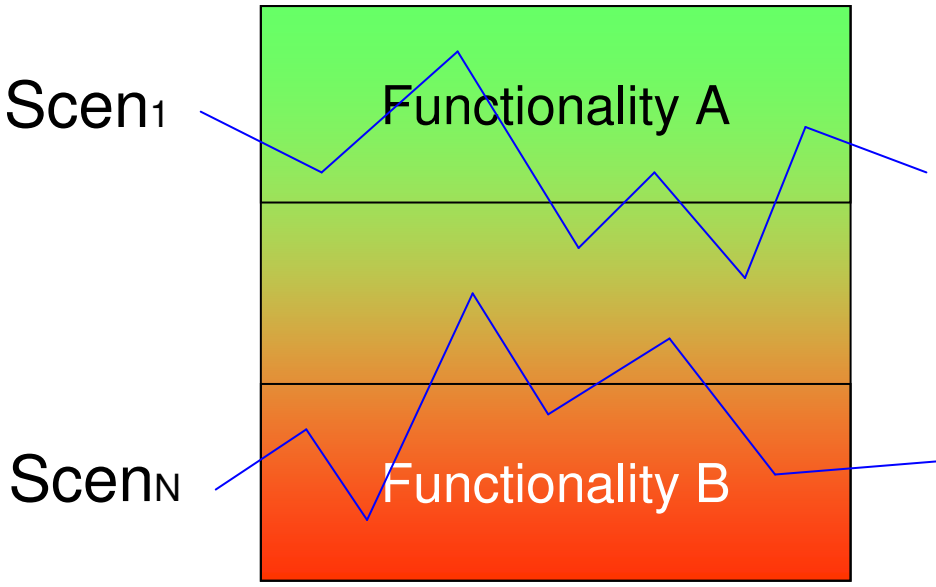
- comprehension
- restructuring
- reengineering

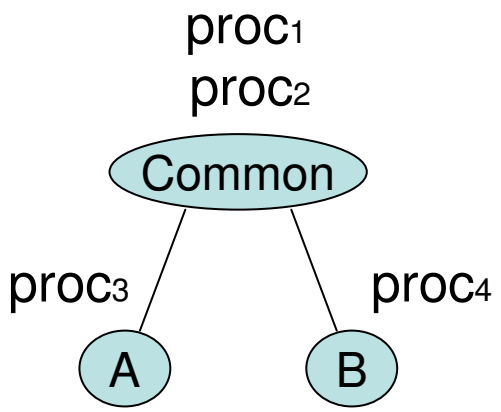


ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

Feature location

Feature location = determine code fragments specific of a given functionality. Beyond **grep** (i.e., lexical search based on regular expressions): dynamic analysis + concept lattice.





Dynamic analysis ➡

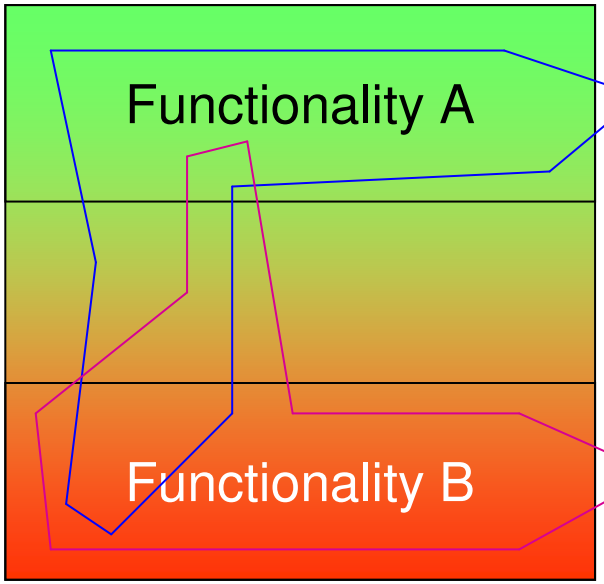
Partial (incomplete) results:

false positives/negatives

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

Slicing

A *slice* is a subprogram that behaves (upon termination) as the original program for the computation of selected variables.



$S(a, n) = \{s_1, \ldots\}$

$S(a, n) \cap S(b, m) \neq \emptyset$

$S(b, m) = \{s_4, \ldots\}$

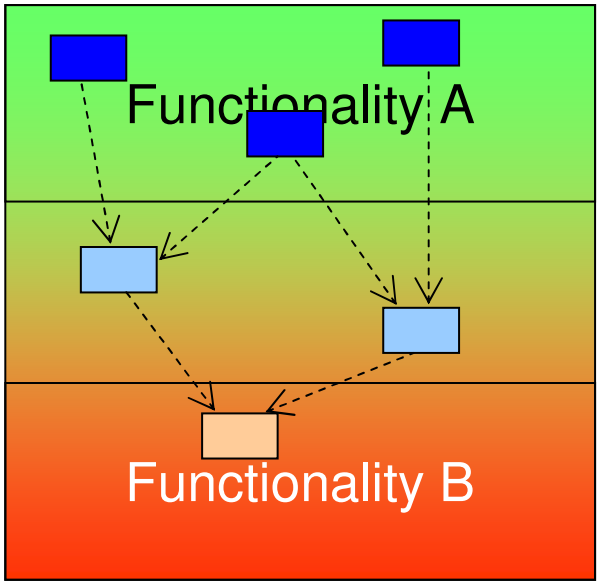
Static analysis ➡

(Over-)conservative results: false positives

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

Impact analysis

Program dependencies are traversed and the change is propagated along them.



- Control dependencies:
  - source = decision statement
  - target = conditionally executed stmts
- Data dependencies:
  - data flows from defs to uses
- Call dependencies

Statically computed dependencies ➡

(Over-)conservative results: false positives

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

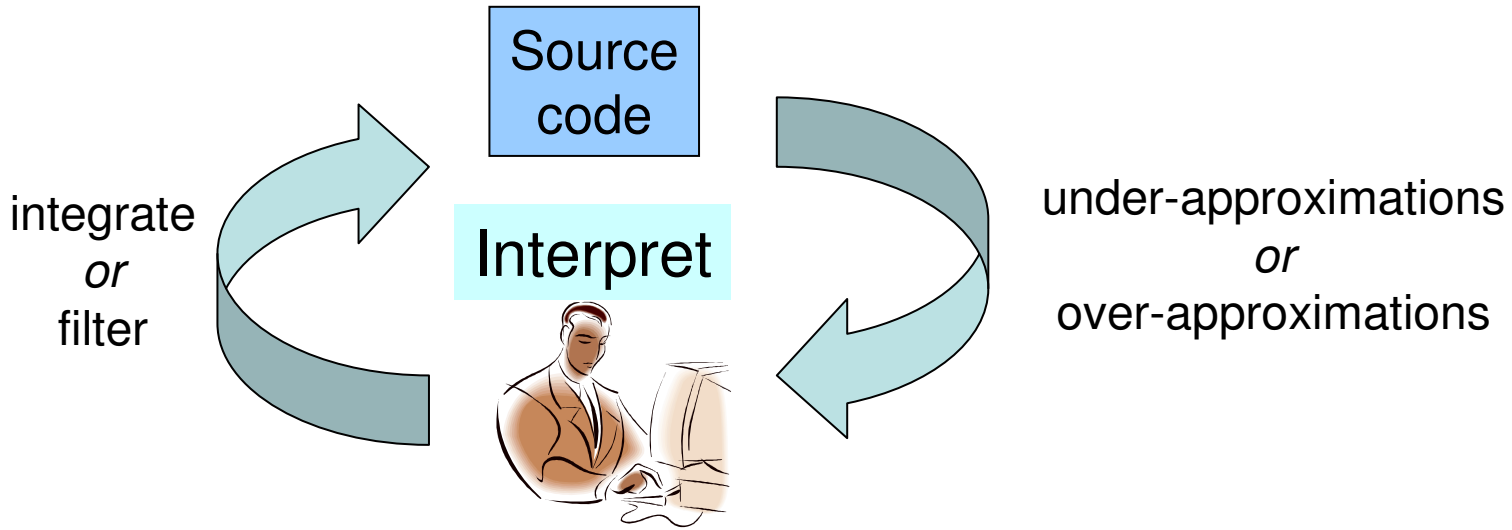
Limitations of reverse engineering

Dynamic analysis:

- Under-approximation of actually involved components.
- Users must integrate it with extra knowledge about the program.

Static analysis:

- Over-approximation of actually involved components.
- Users must filter it with extra knowledge about the program.



ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

Reverse engineering for software tampering

Bad news:

- Feature location, slicing and impact analysis can be automated.
  - Previously acquired knowledge can be reused, if the same or similar authentication mechanisms are employed.

Good news:

- Humans are in the loop.
  - Obfuscation and similar techniques are effective as long as they make human understanding of the code harder.

Mutation testing

A program can be delivered only when test cases succeed revealing a set of known bugs (killing all mutants).

```
1   program P
2   begin
3      input(x);
4      if (x > 0)
5           x++;
6      end if
7      print(x);
8   end program
```

4	`if (x == 0)`	M1
4	`if (x != 0)`	M2
4	`if (x >= 0)`	M3
5	`x--;`	M4
5	`x = 0;`	M5
5	`x = 1;`	M6

```
x = 1
```

⇩

Kills: M1, M4, M5, M6

```
x = 0
x = 1
```

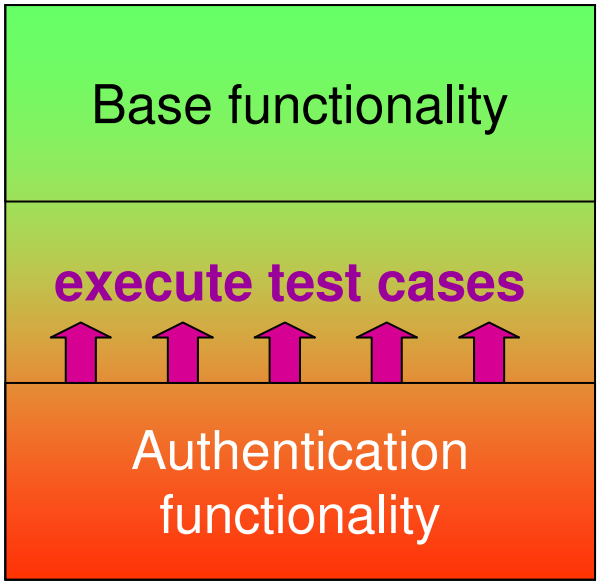
⇩

Kills: M1, M3, M4, M5, M6

Both test suites are not satisfactory because they cannot reveal the defect in **M2**.

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

Mutation analysis for software integrity



1. Mutation adequate test cases are defined.
2. The authenticator module runs periodically the test cases and compares the actual output against the expected one.
3. When mutations are revealed by the test cases, authenticity tags are no longer generated.

Being based on the I/O relationship of the base functionality, this technique is potentially resistant to reverse engineering attacks that change it (impact set = whole program).

ITC
irst
CENTRO PER LA RICERCA
SCIENTIFICA E TECNOLOGICA

Conclusions

- Reverse engineering represents a serious threat to software integrity and authentication.

- Existing applications of reverse engineering indicate that the process can be only partially automated and the human is necessarily in the loop.

- Mutation analysis can be employed to generate authenticator modules that are able to reveal (malicious) software mutations.