ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SPaCIoS

Semi-valid Input Coverage for Fuzz Testing

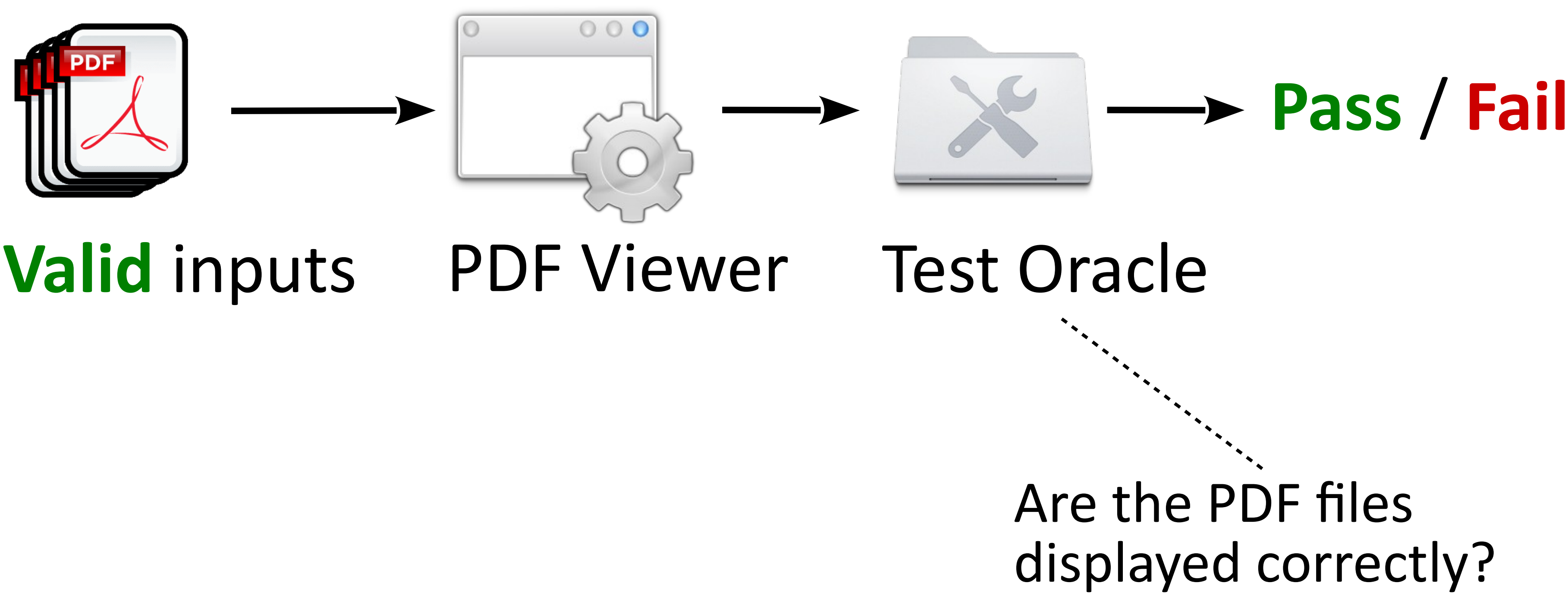
Petar Tsankov, Mohammad Torabi Dashti, David Basin

Institute of Information Security

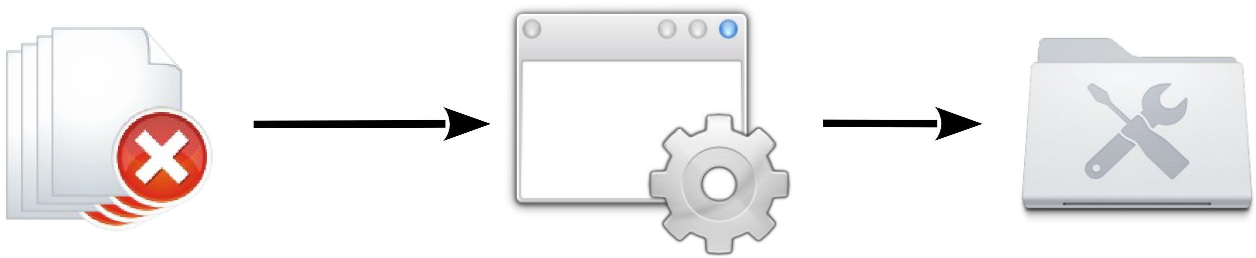
ETH Zurich

Fuzz Testing

Testing a PDF Viewer

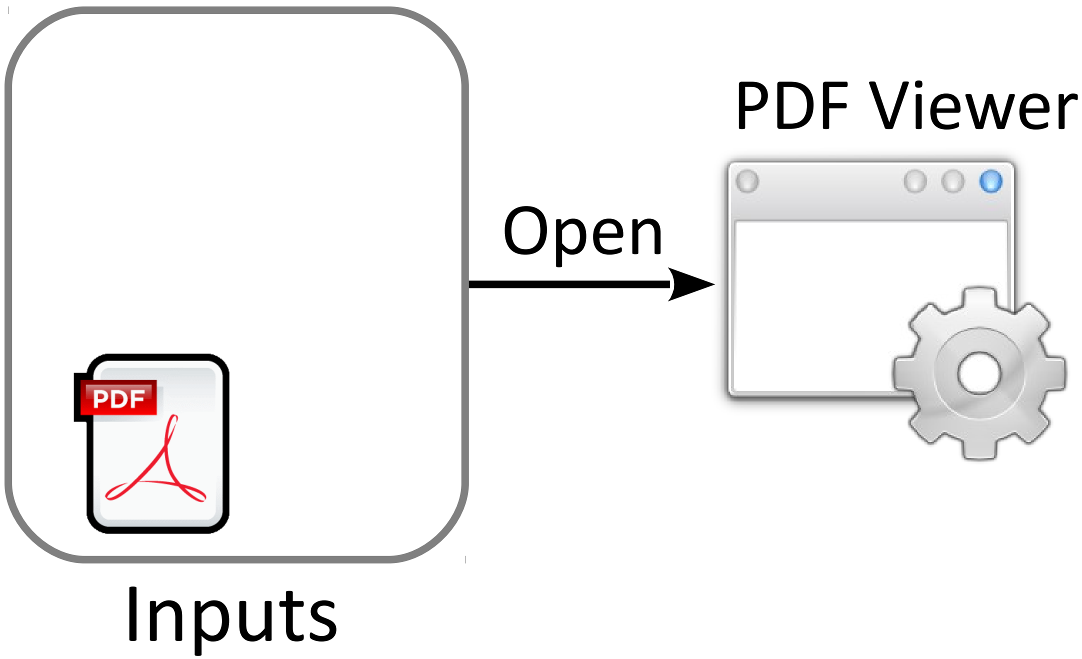


Fuzz-testing a PDF viewer testing

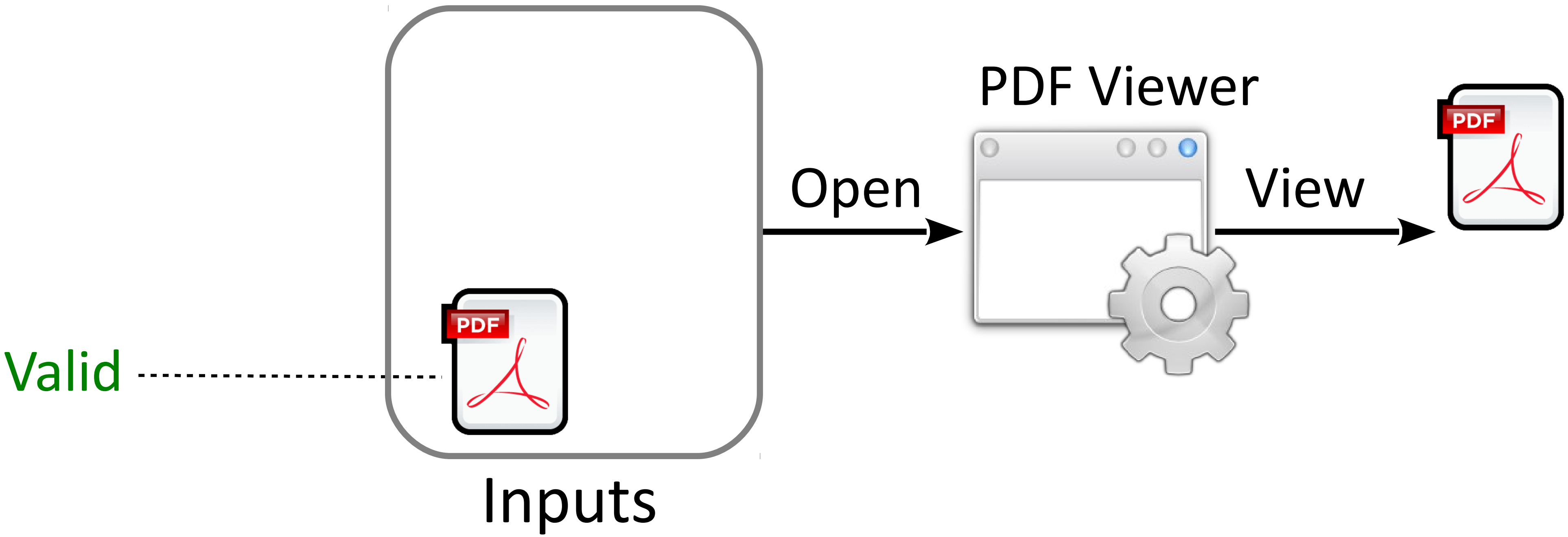


Are there any **security faults**?
(e.g. memory errors)

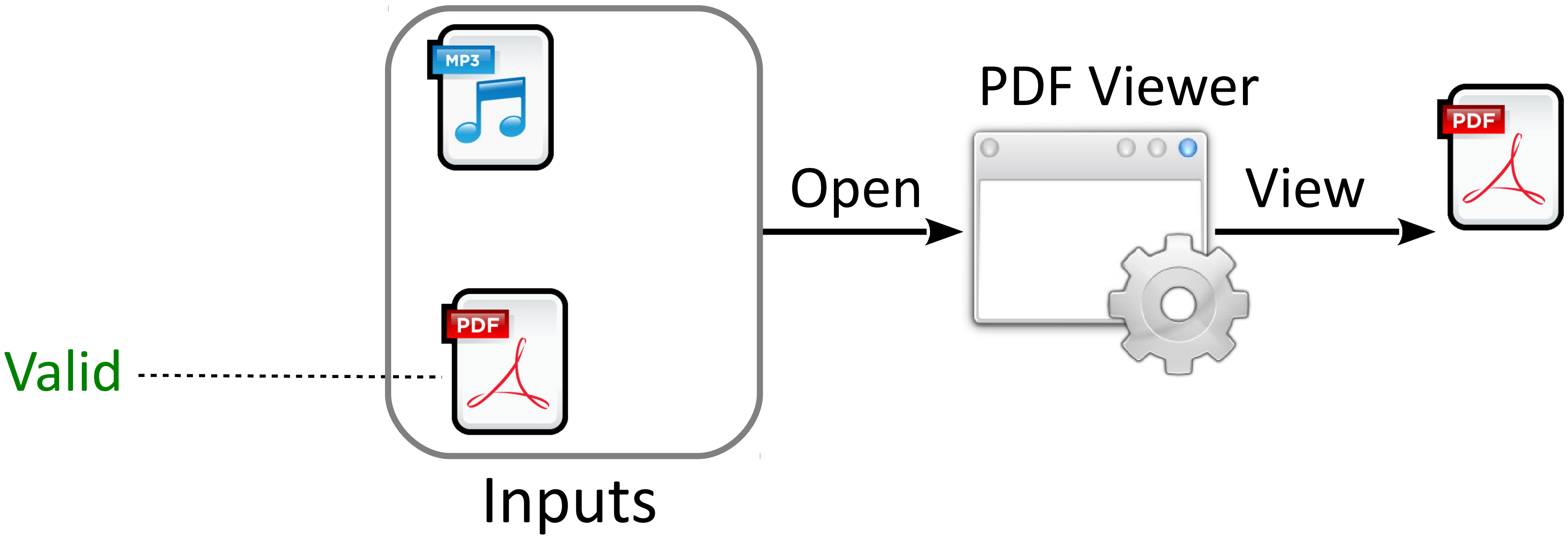
Semi-valid Inputs



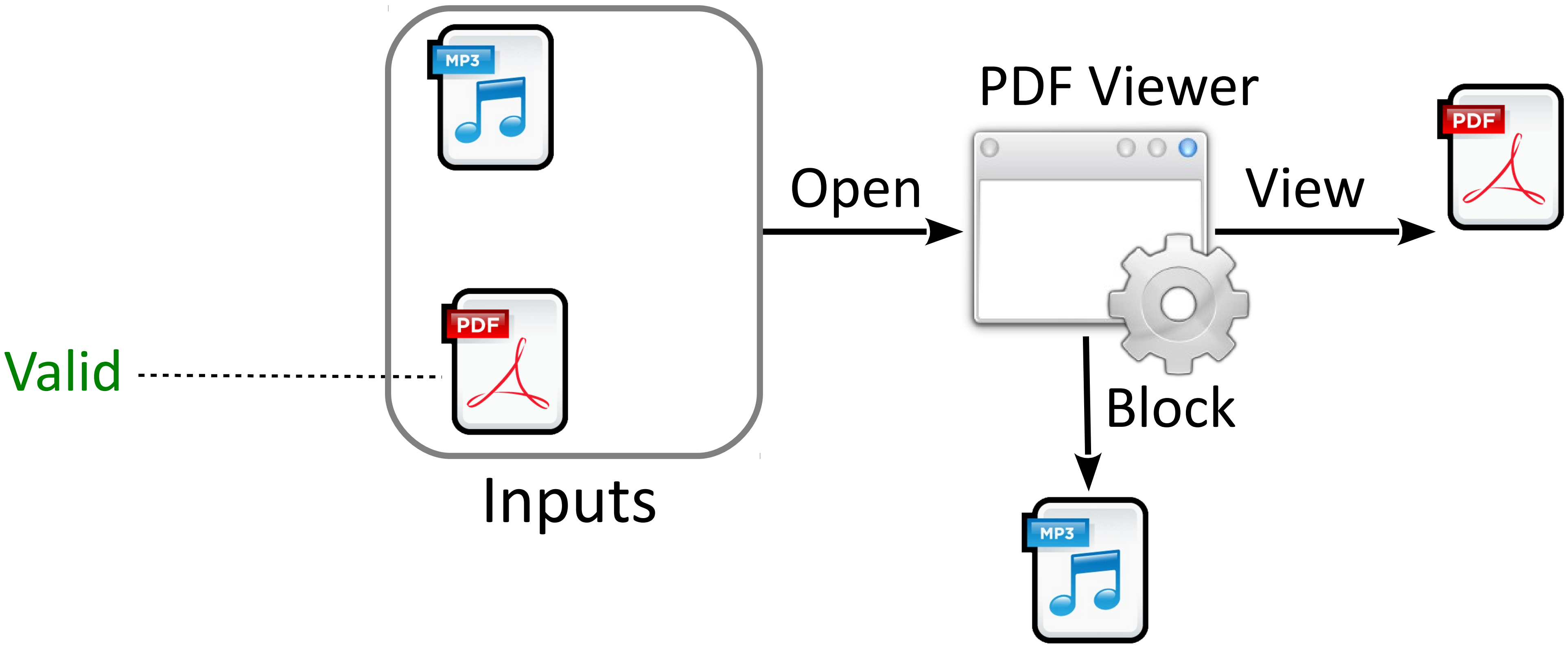
Semi-valid Inputs



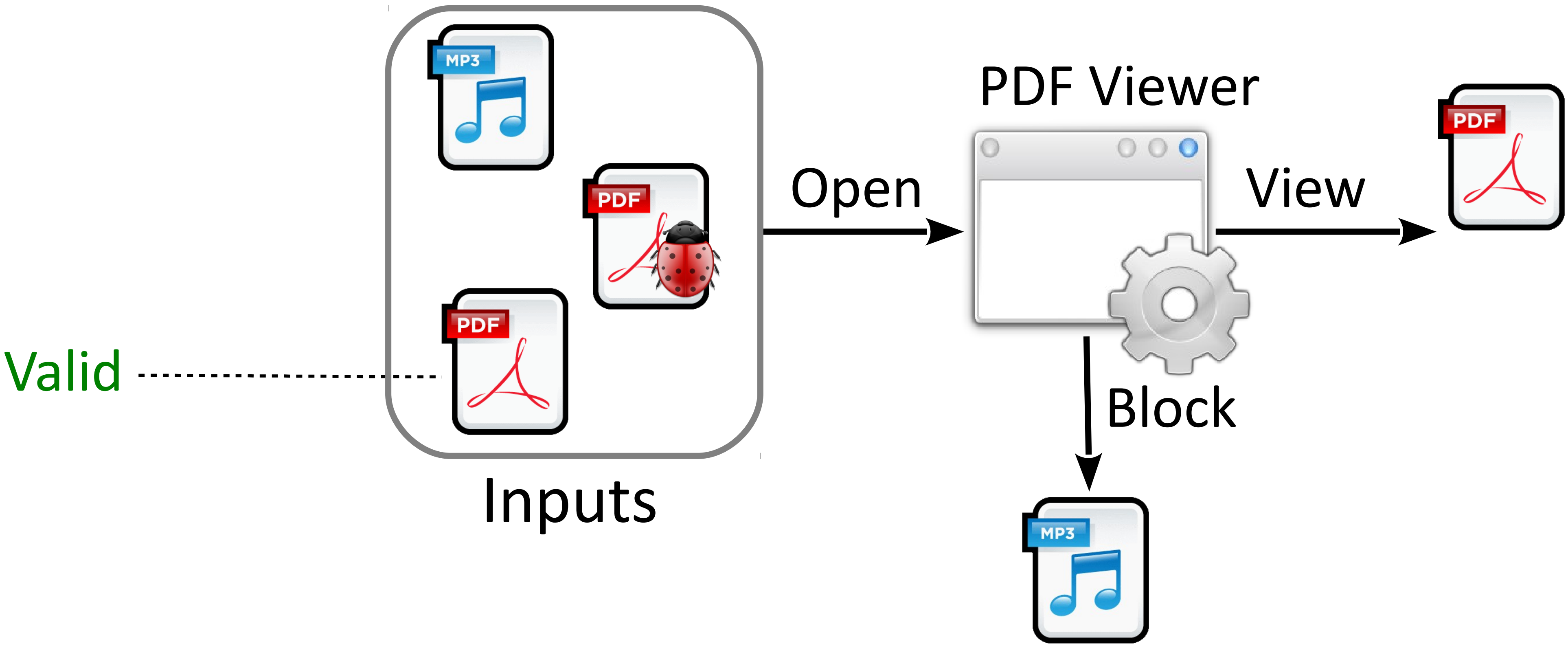
Semi-valid Inputs



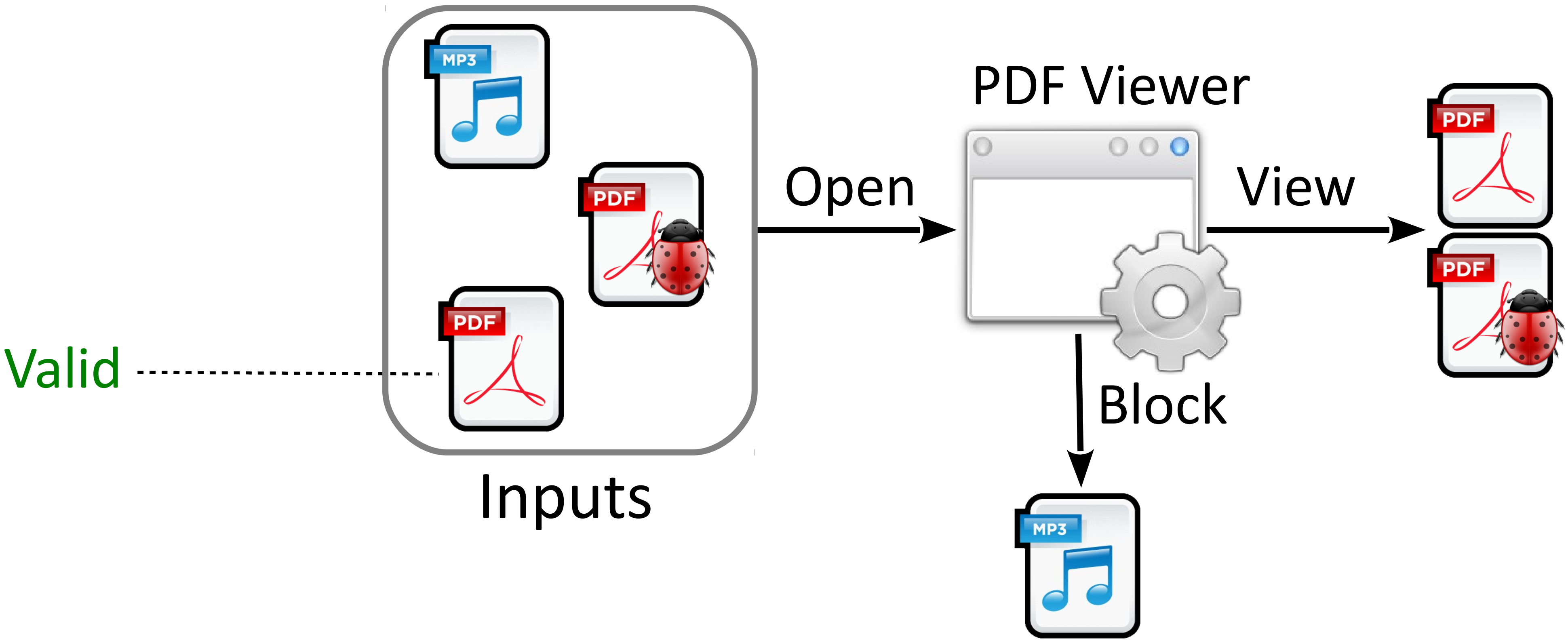
Semi-valid Inputs



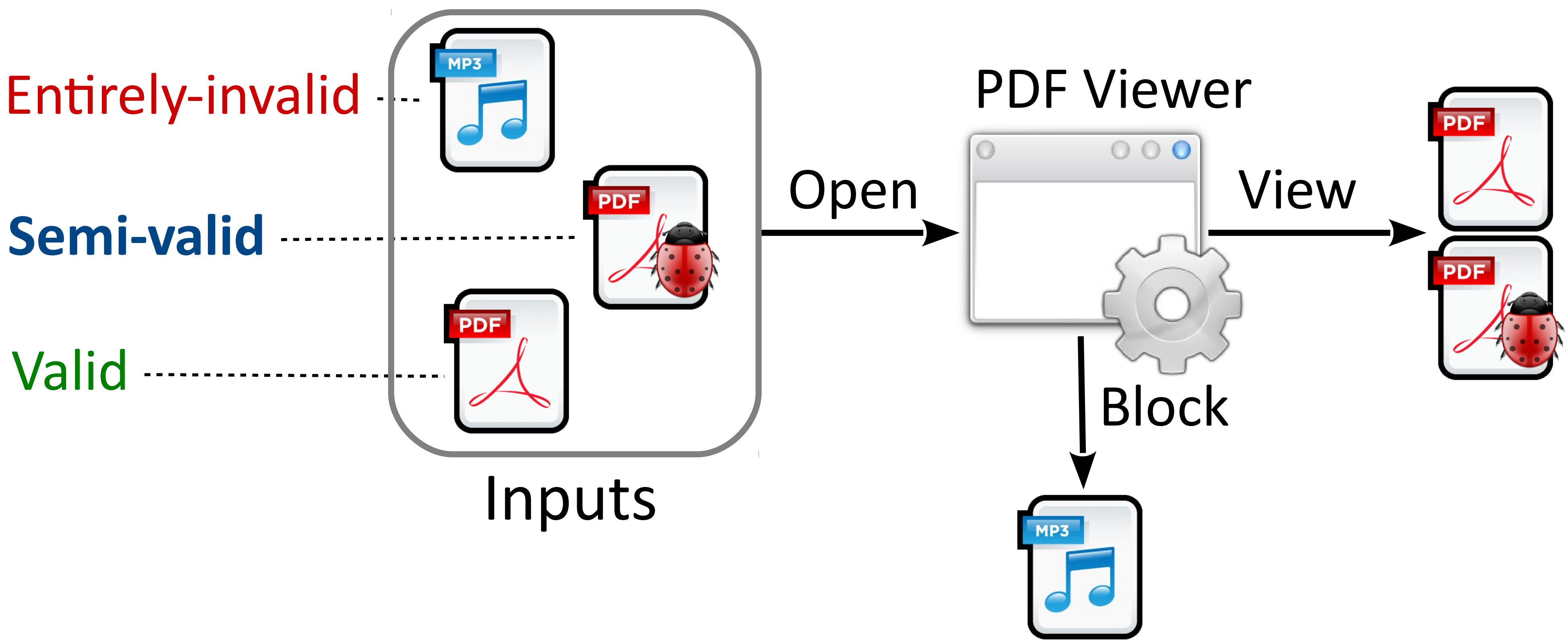
Semi-valid Inputs



Semi-valid Inputs

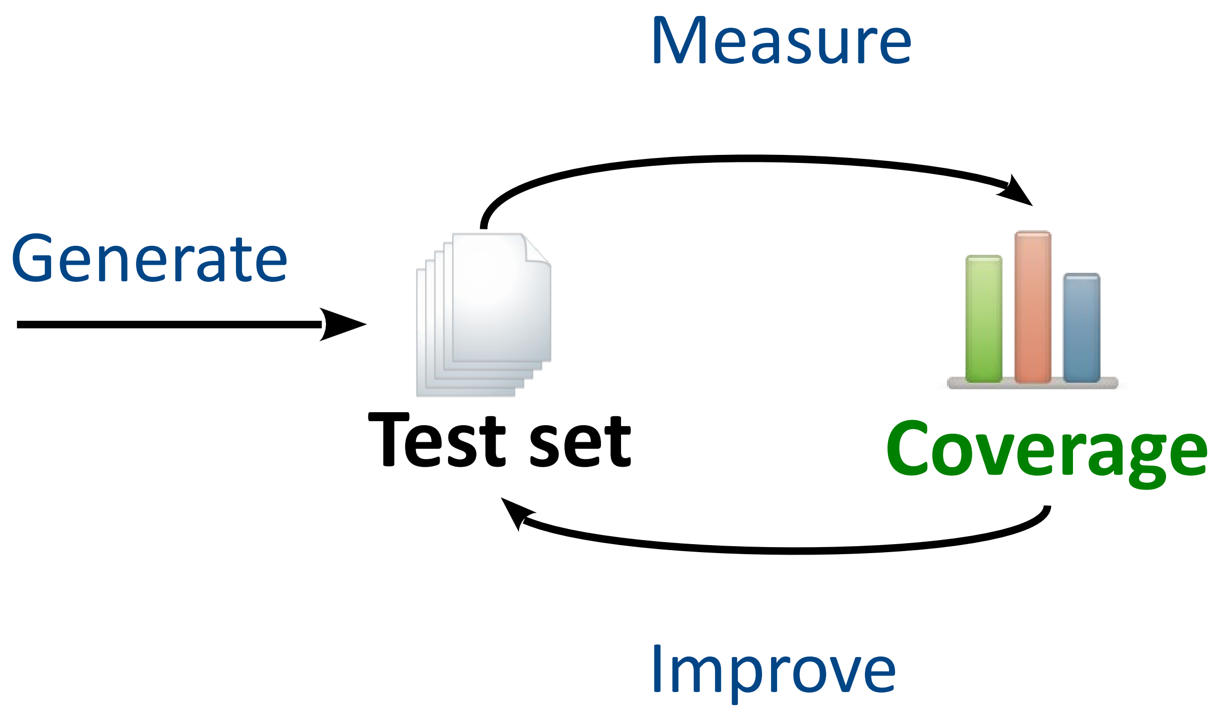


Semi-valid Inputs



- Entirely-invalid inputs get blocked.
- **Semi-valid inputs** are essential for fuzz testing.

Coverage Criteria



- Low coverage hints at missing test cases.
- No existing coverage metric tailored to fuzz testing.
  - existing metrics do not tell us how thoroughly we have tested with semi-valid inputs.

Coverage for Fuzz Testing

Semi-valid Input Coverage (**SVCov**)

- Constraints define whether an input is valid or not.

  "The third byte is the XOR of the first two bytes." (**C1**)

Semi-valid Input Coverage (**SVCov**)

- Constraints define whether an input is valid or not.

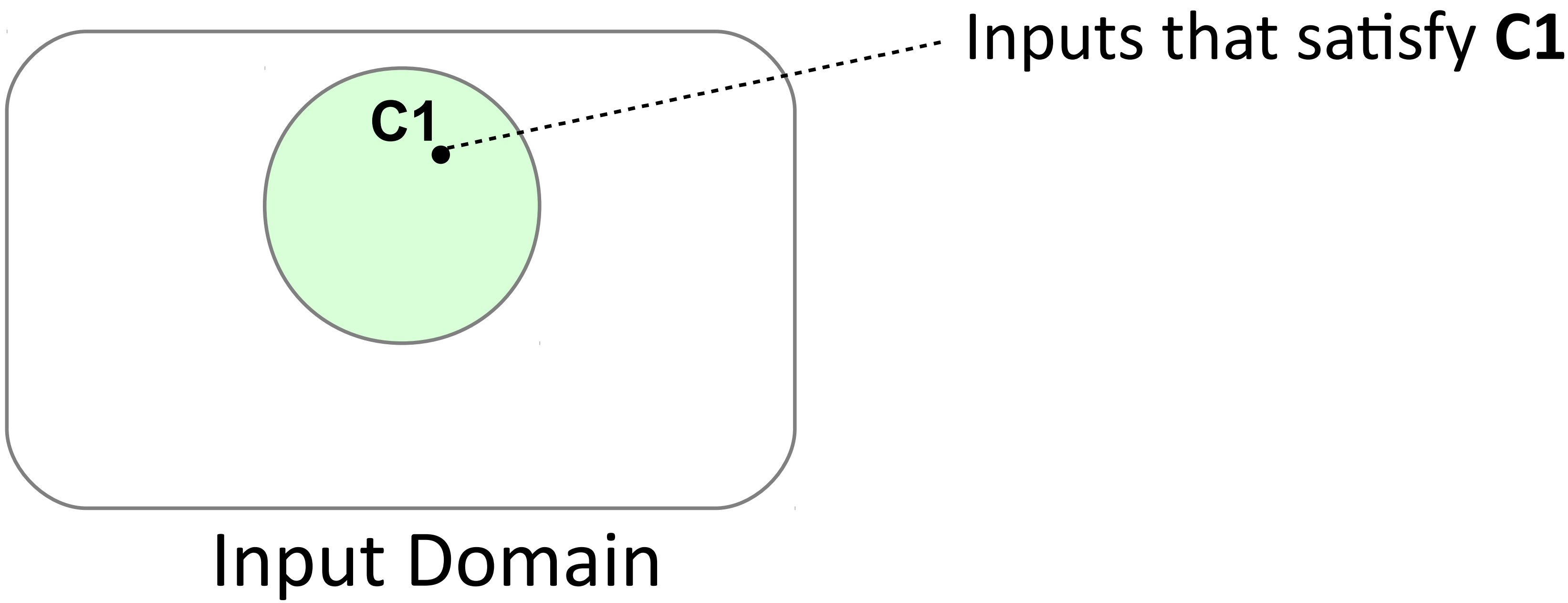
  “The third byte is the XOR of the first two bytes.” (**C1**)

Input Domain

Semi-valid Input Coverage (**SVCov**)

- Constraints define whether an input is valid or not.

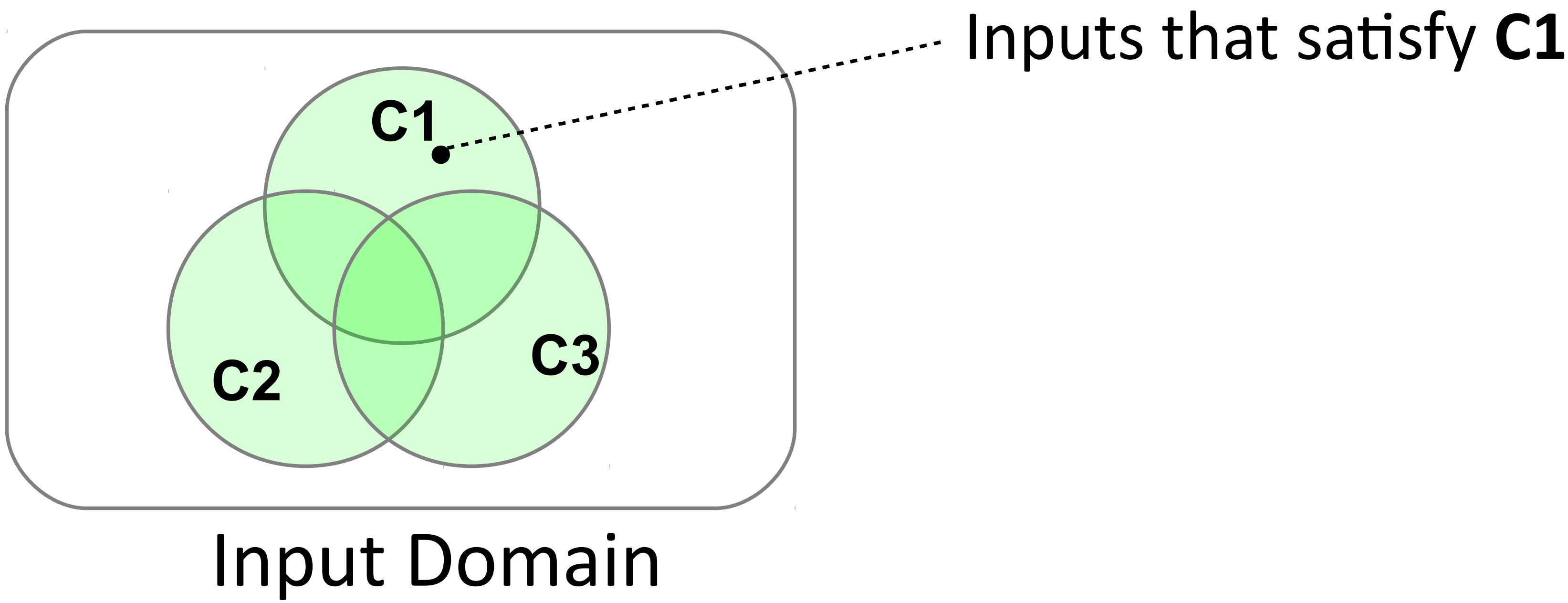
  "The third byte is the XOR of the first two bytes." (**C1**)



Semi-valid Input Coverage (**SVCov**)

- Constraints define whether an input is valid or not.

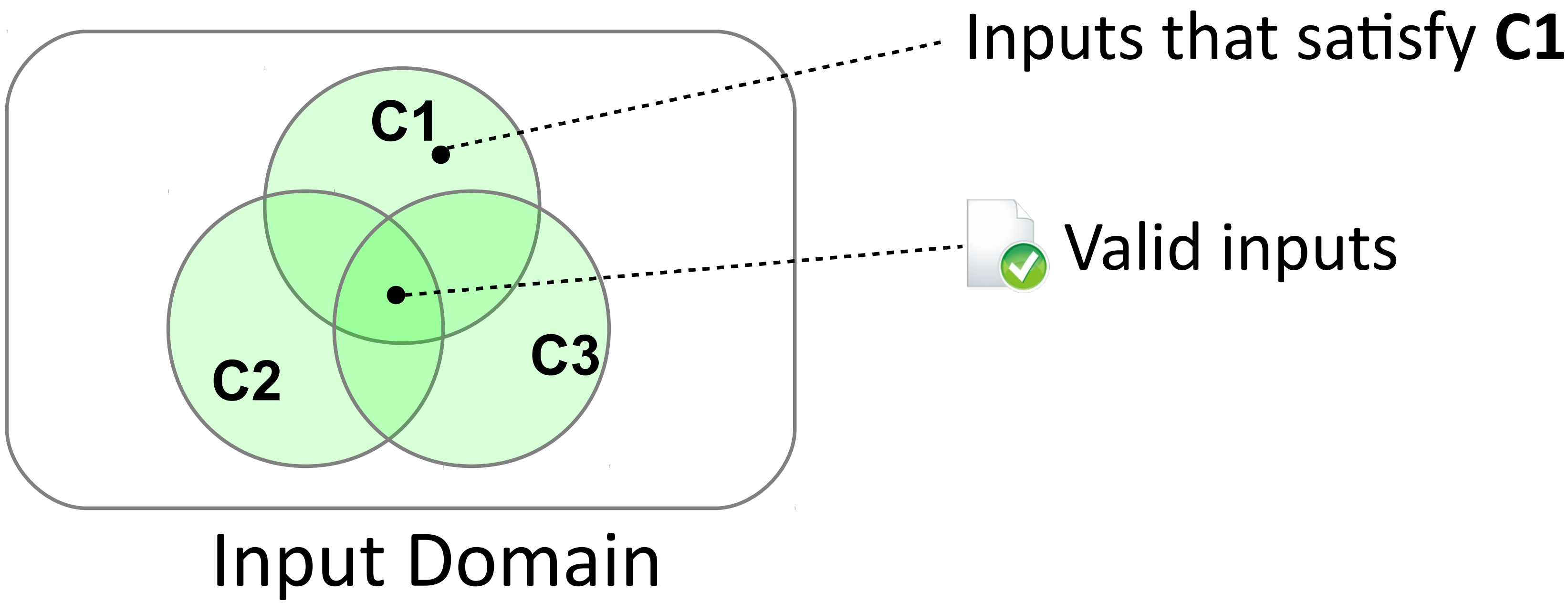
  "The third byte is the XOR of the first two bytes." (**C1**)



Semi-valid Input Coverage (**SVCov**)

- Constraints define whether an input is valid or not.

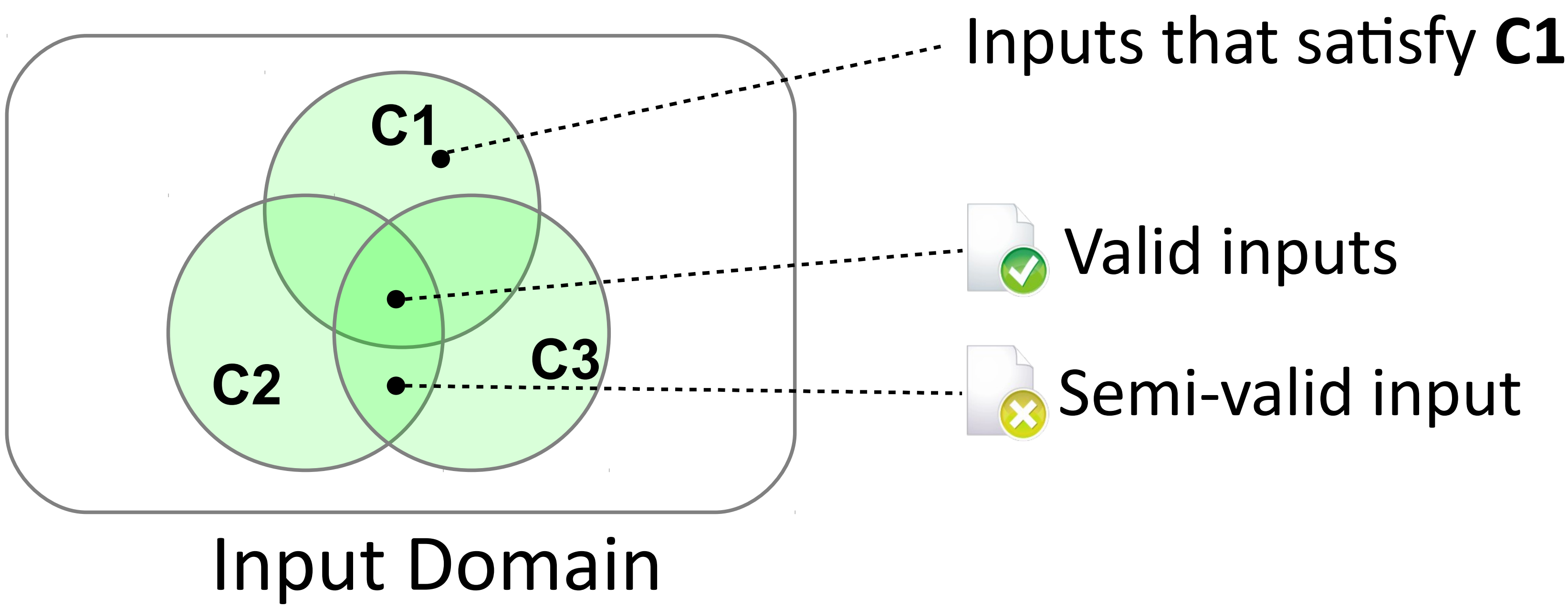
  "The third byte is the XOR of the first two bytes." (**C1**)



Semi-valid Input Coverage (**SVCov**)

- Constraints define whether an input is valid or not.

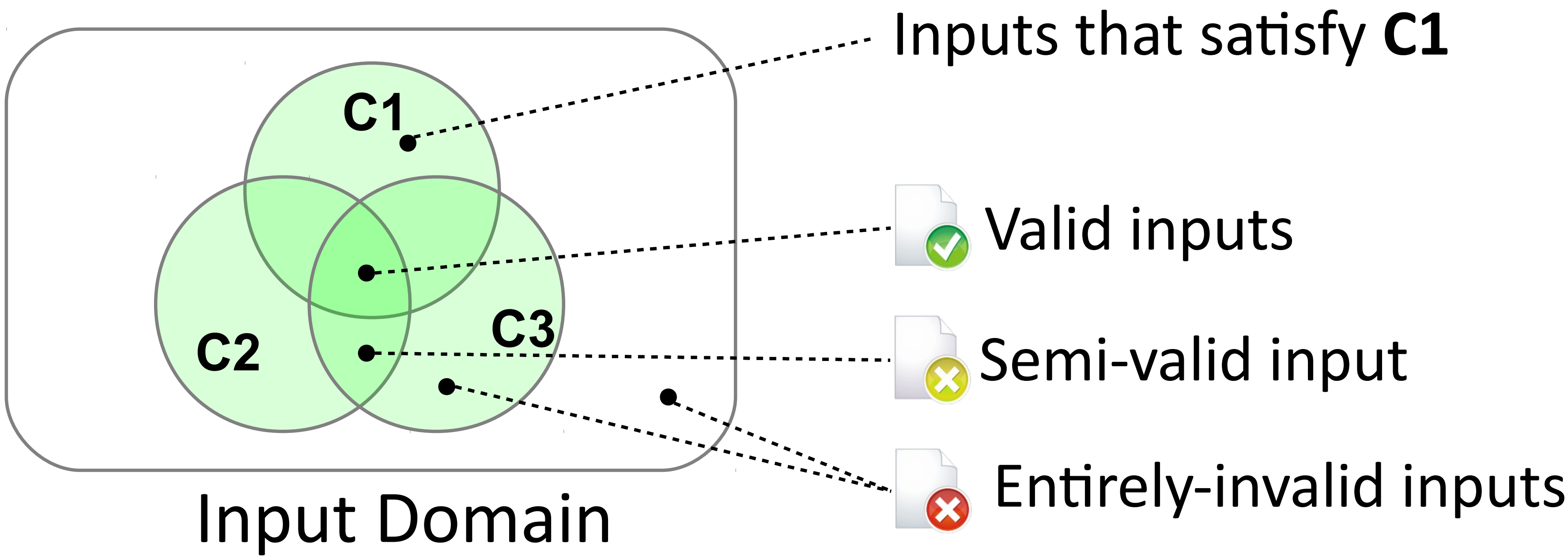
  "The third byte is the XOR of the first two bytes." (**C1**)



Semi-valid Input Coverage (**SVCov**)

- Constraints define whether an input is valid or not.

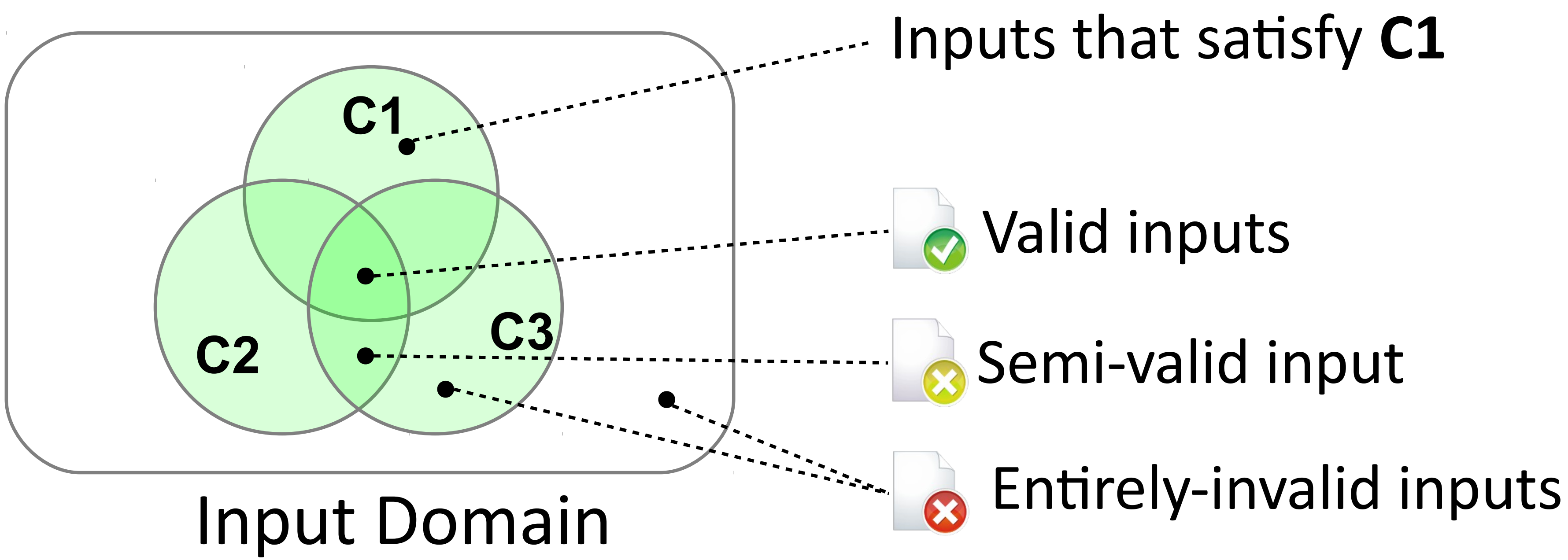
  "The third byte is the XOR of the first two bytes." (**C1**)



Semi-valid Input Coverage (**SVCov**)

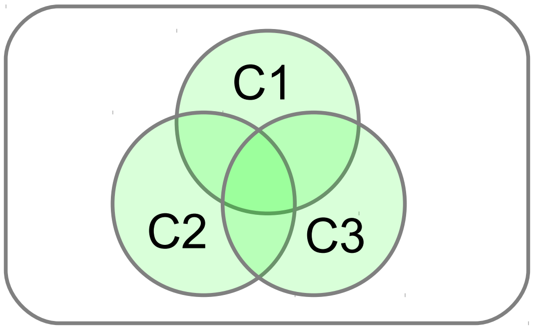
- Constraints define whether an input is valid or not.

"The third byte is the XOR of the first two bytes." (**C1**)



$$\textbf{SVCov} = \frac{\text{\# covered semi-valid partitions}}{\text{\# total semi-valid partitions}}$$

SVCov Properties

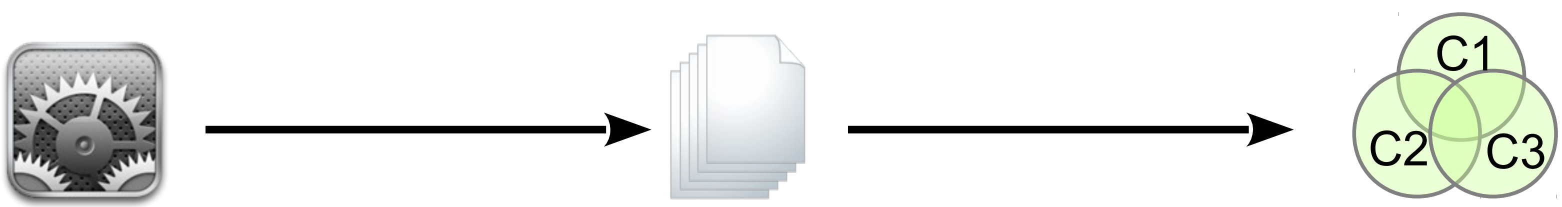


$$\text{SVCov} = \frac{\text{\# covered semi-valid partitions}}{\text{\# total semi-valid partitions}}$$

- ✓ Independent to test generation method.
- ✓ Valid inputs do not contribute to SVCov.

- ⚠ The usefulness of SVCov depends on the constraints.
- ⚠ 100% SVCov does not guarantee that the tests reveal all faults.

Using SVCov



Fuzzing tool

Test set

SVCov

Using SVCov

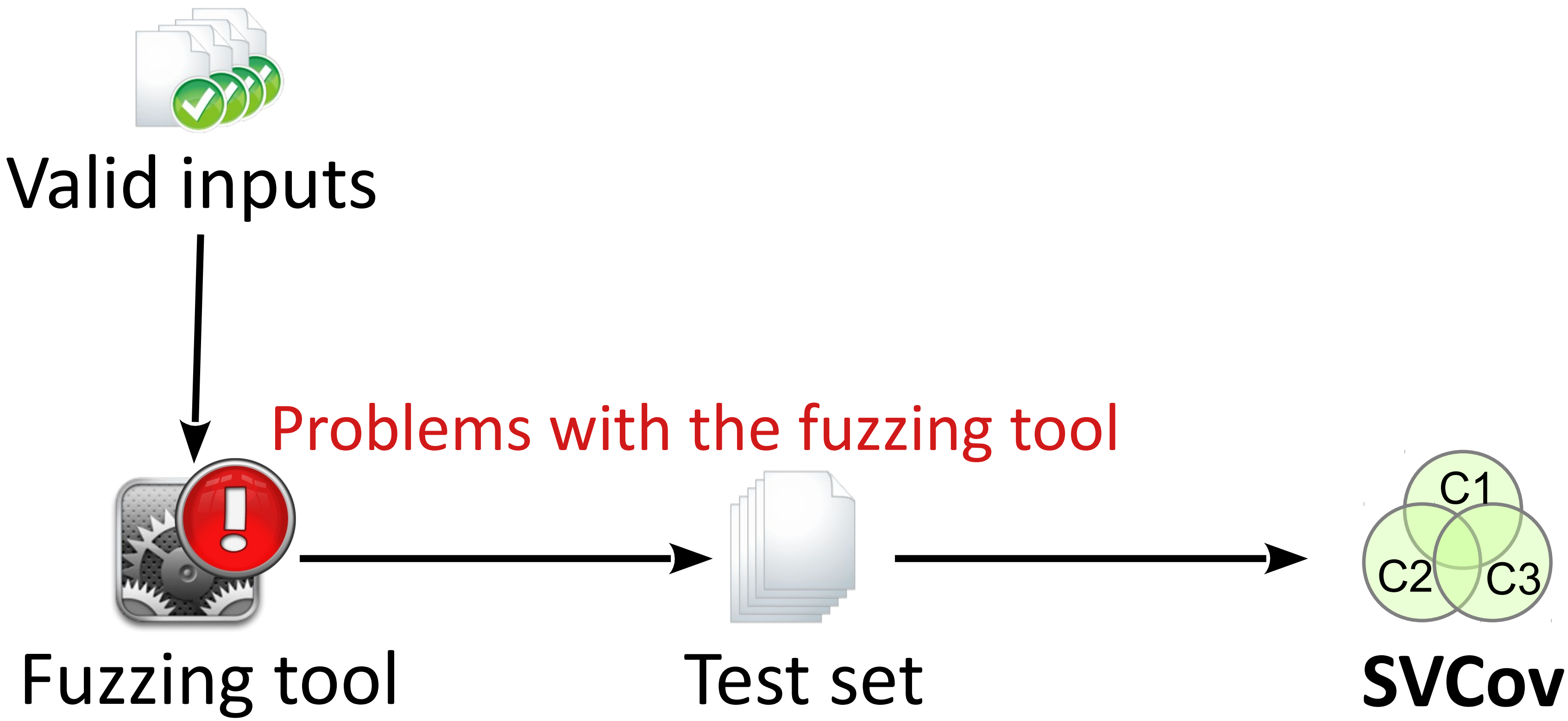
Problems with the fuzzing tool

Fuzzing tool

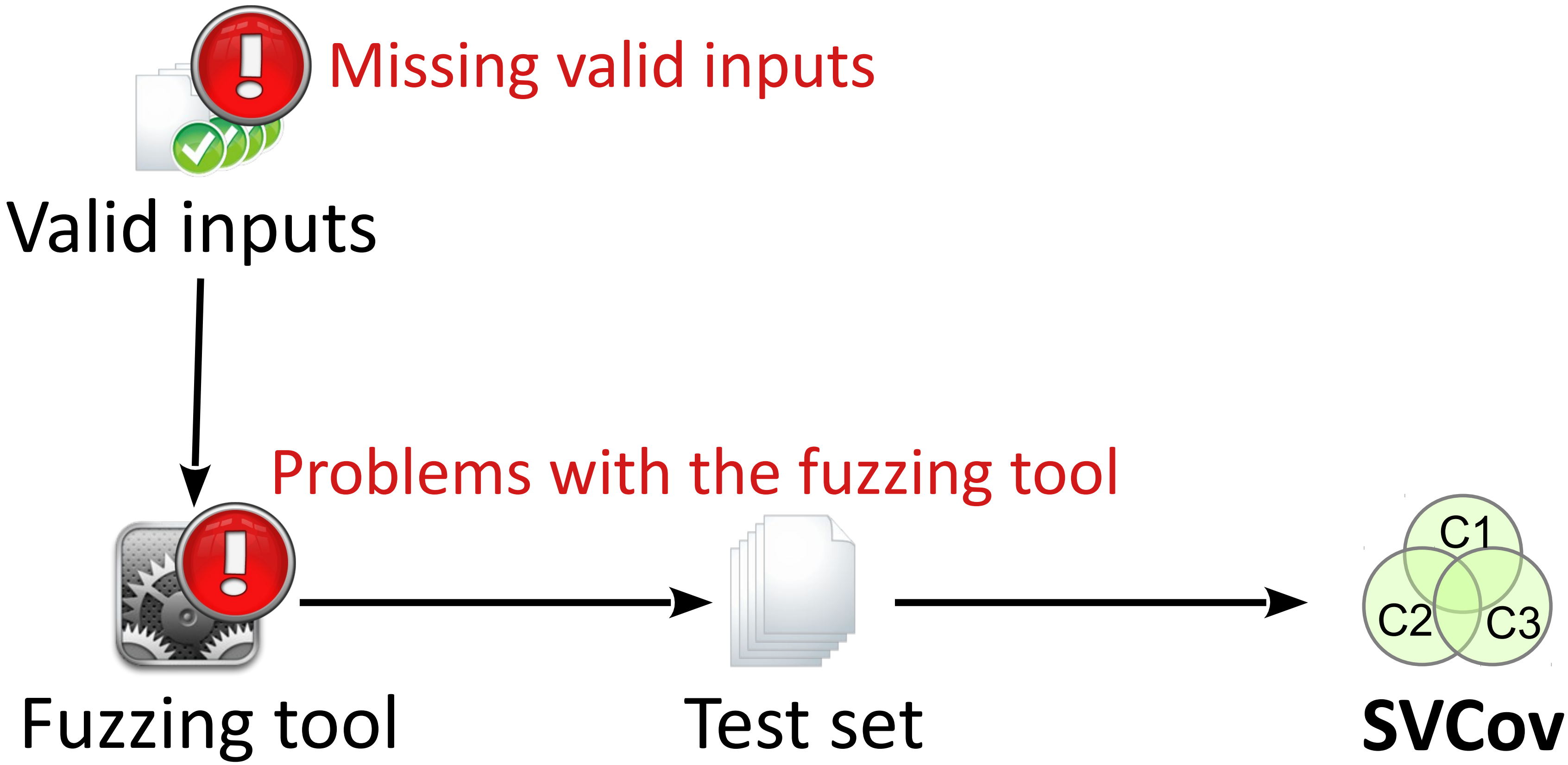
Test set

SVCov

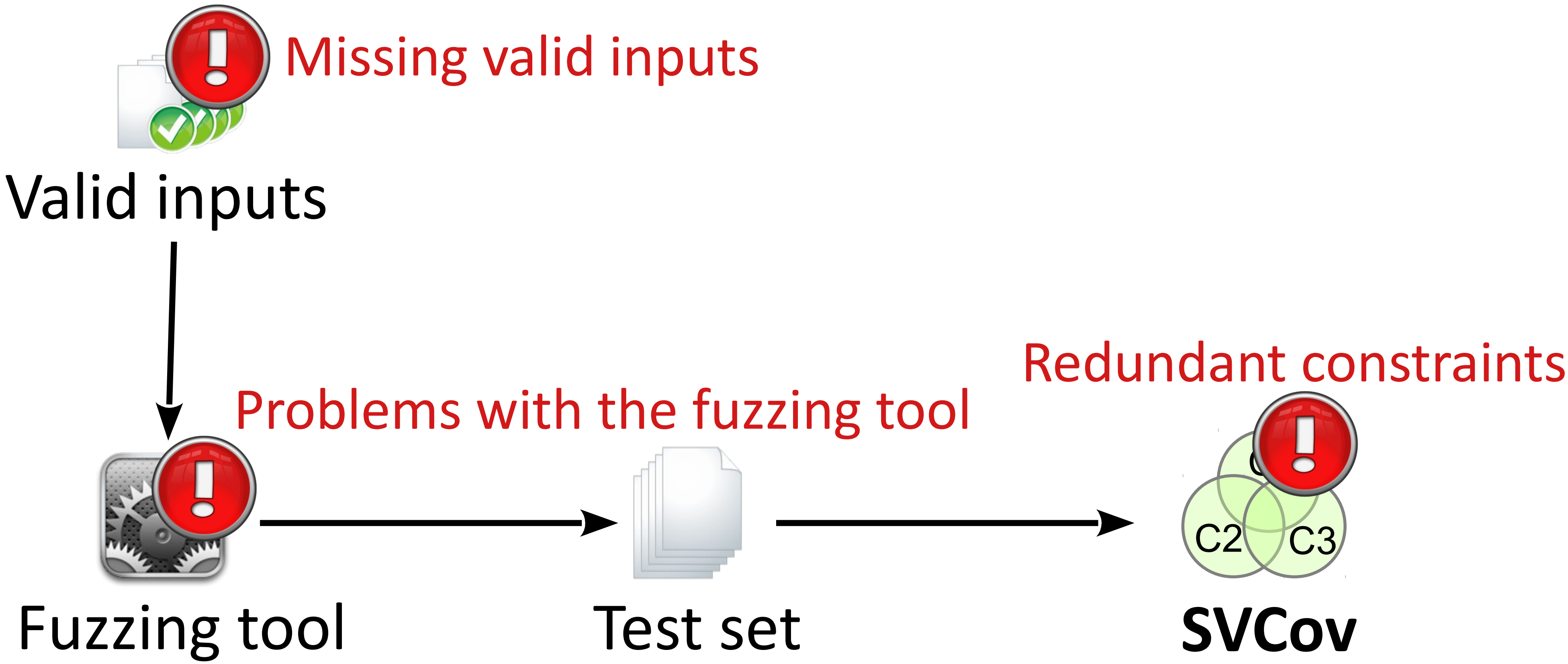
Using SVCov


Valid inputs
Problems with the fuzzing tool
C1
C2
C3
Fuzzing tool
Test set
SVCov

Using SVCov


Missing valid inputs
Valid inputs
Problems with the fuzzing tool
Fuzzing tool
Test set
C1
C2
C3
SVCov

Using SVCov


Missing valid inputs
Valid inputs
Problems with the fuzzing tool
Redundant constraints
C2
C3
Fuzzing tool
Test set
SVCov

Case Study

Case Study

Research questions:

- **RQ1: Feasibility**

  Can we precisely define the semi-valid inputs of the SUT and efficiently measure SVCov?

- **RQ2: Relevance to coverage**

  Does measuring SVCov provide meaningful information on how to improve a test set's coverage?

- **RQ3: Relevance to discovering faults**

  Does increasing SVCov result in discovering additional faults?

Case Study: Artifacts

- Test subject: OpenSwan
  – IKE implementation for Linux, 600K LOC.
  – Input specification: RFC2407, RFC2408, RFC2409.
- Fuzzing tool: SecFuzz
  – Mutation-based fuzzer for security protocols.
- Test oracle: MemCheck
  – Detects memory errors.
- SVCov checker
  – Currently supports only IKE.

RQ1: Feasibility

- We focused on "must (not) sentences" in the RFCs:

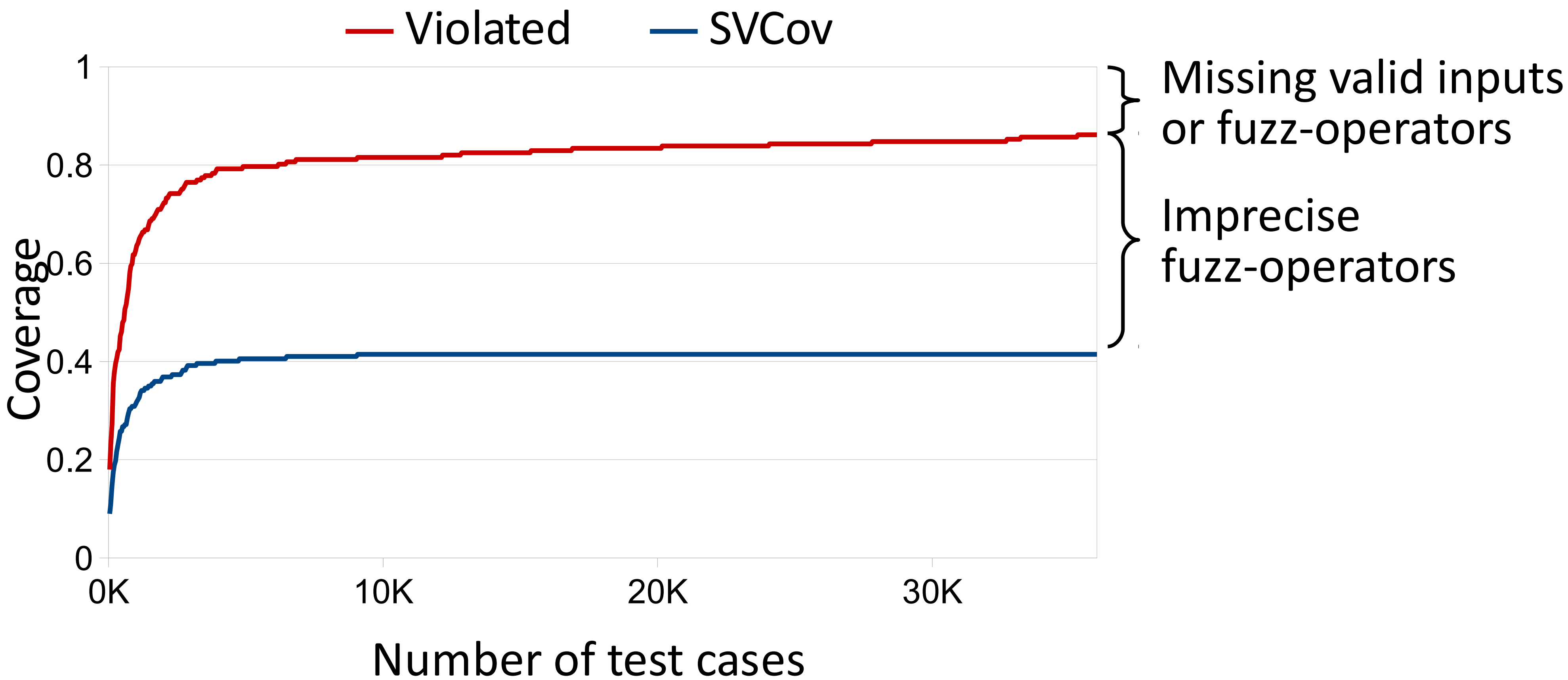
  *"If a message contains a proposal payload, then the proposal's next-payload field **must** be set to 2 or 0."*

- The specification of constraints for IKE is straightforward:
  - Number of constraints: **217**.
  - Time to extract the constraints: **8 person hours**.

- Negligible overhead for measuring SVCov:
  - Time to check all constraints for each test case: **41 ms**.
  - Time to execute a test case: **1000 ms**.

RQ2: Relevance to Coverage

SVCov (initial)

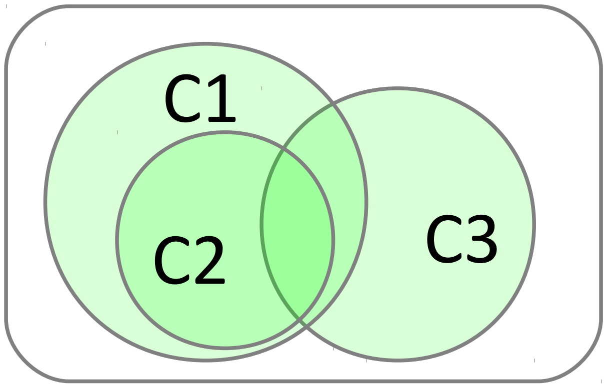


- Many constraints are violated, but not uniquely.
- Some constraints are never violated.

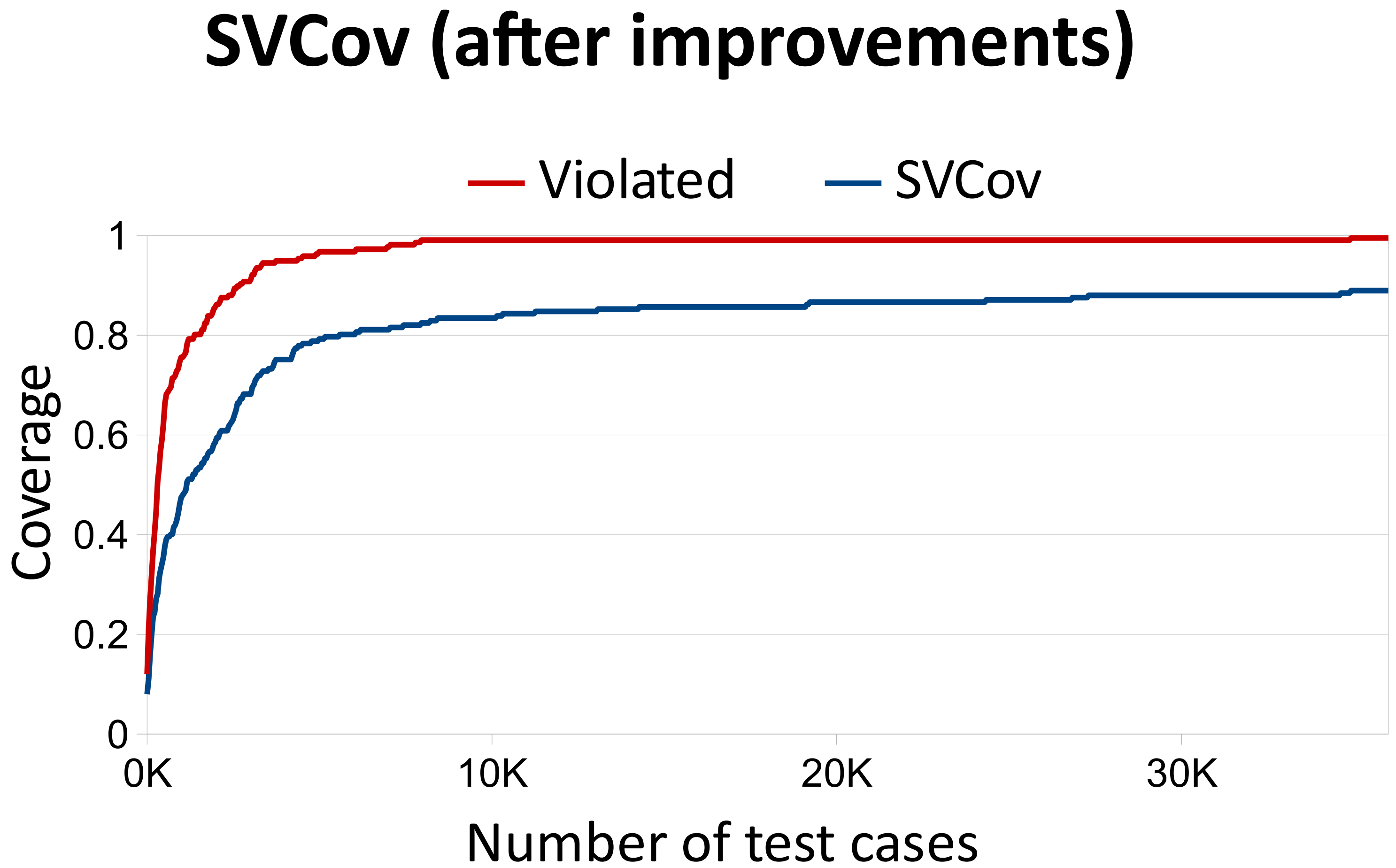
RQ2: Relevance to Coverage

SVCov analysis

- Problems in the fuzzing tool
  - Imprecision in the “insert payload” fuzz operator.
  - Insert random numbers limited to [0, 100].
  - ...
- Missing valid inputs
  - No valid inputs for IPv6 and ASN.1 X500 DN.
- Redundant constraints



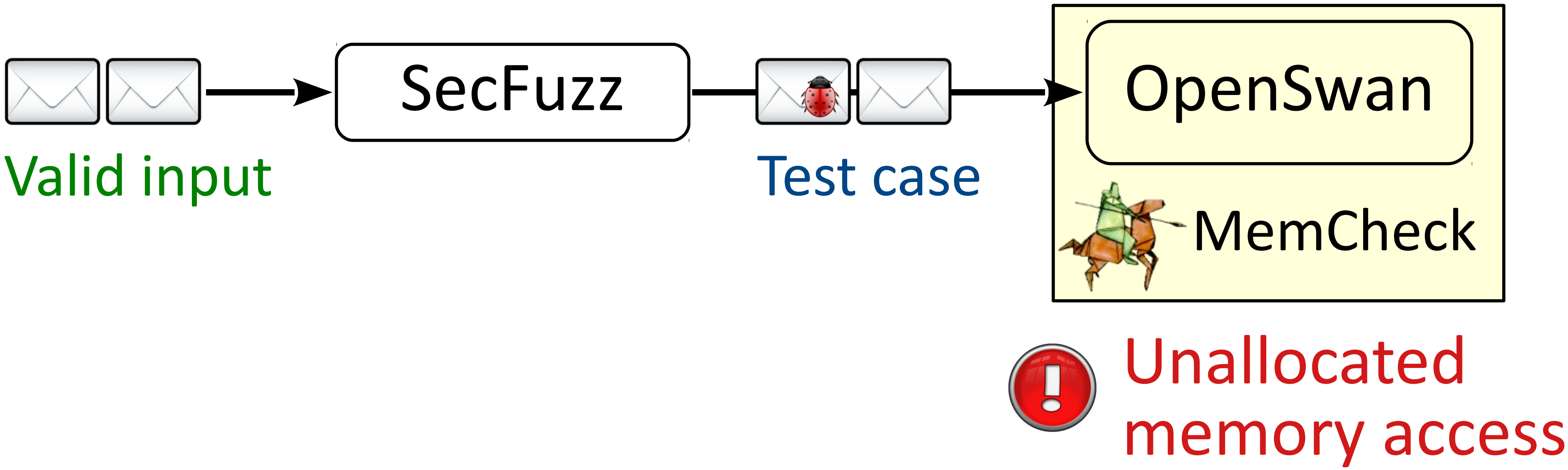
RQ2: Relevance to Coverage



- SVCov improved from 41% to 89%.
- All constraints are violated.
- 9% of the constraints are not uniquely violated.

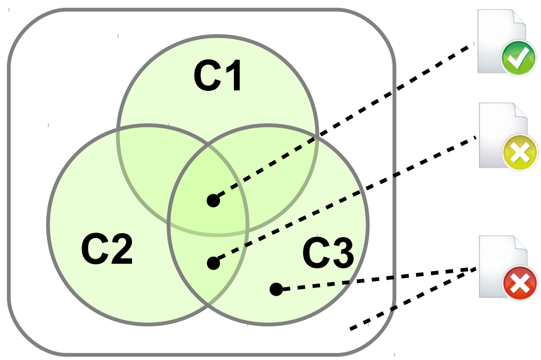
RQ3: Relevance to Discovering Faults

- A previously unknown security fault revealed after improving SVCov.



- The valid input was missing in the first experiment.
- The test case belongs to a semi-valid partition.

SVCov Contributions



Easy-to-use coverage for fuzz testing

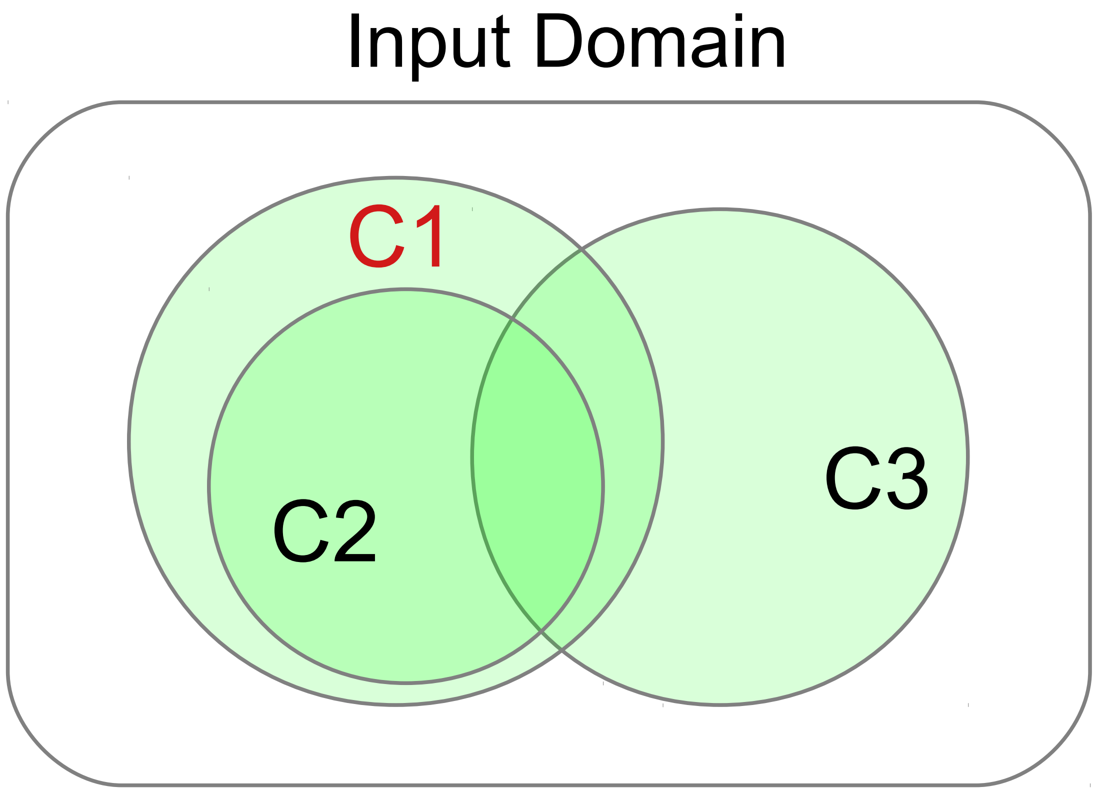
Independent of the fuzz-testing technique

Pinpoint subtle problems in fuzz testing

Promising initial empirical results

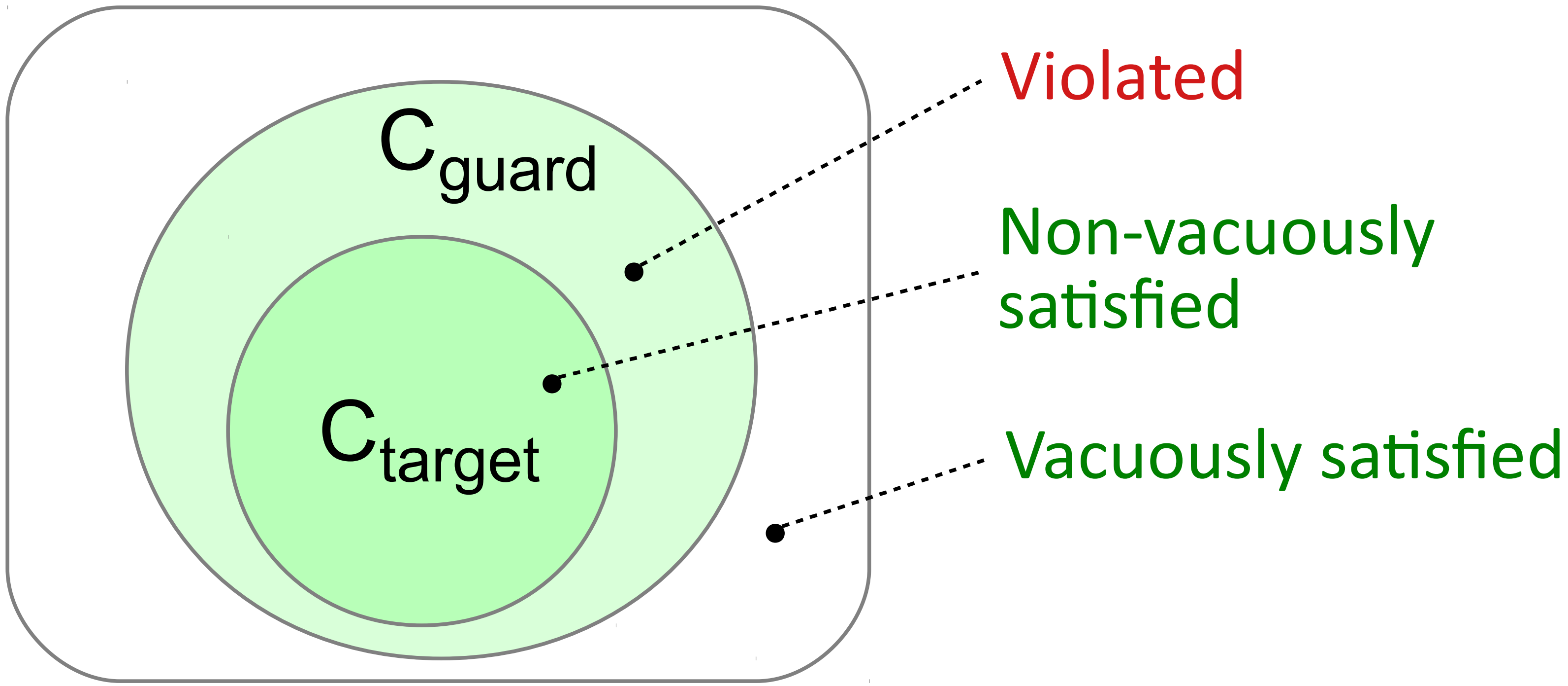
Backup Slides

Redundant Constraints



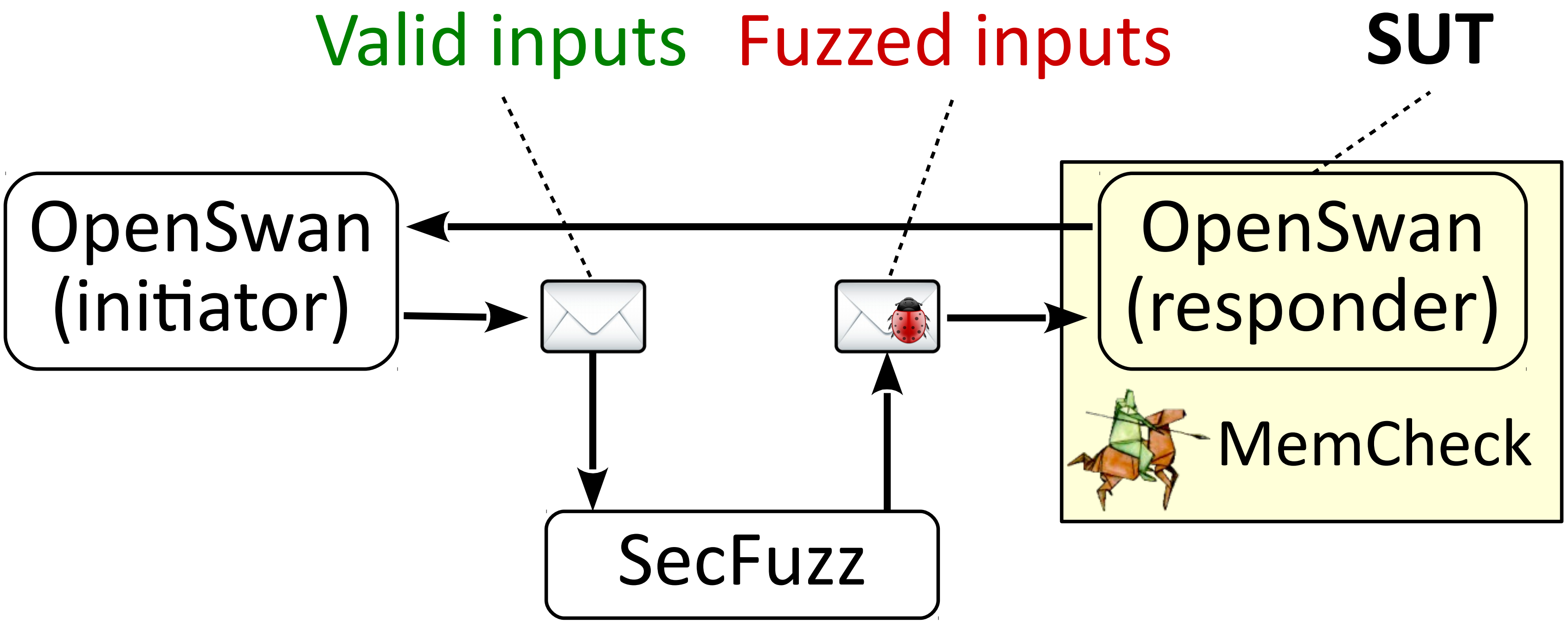
- Constraint C1 is redundant.
  - removing C1 does not change the set of valid inputs.
- Constraint C1 cannot be uniquely violated.
  - Any input that violates C1 also violates C2.

Missing Valid Inputs



- To violate a constraint we need an input that satisfies the constraint non-vacuously.

Case Study: Setup



- We measure and report SVCov of the fuzzed inputs.
- Measure SVCov of the valid inputs to check for missing inputs.