Semi- automatic tool for test cases generation on X.509 parser

REDOCS 2022

Ferran Alborch • Alba Martínez • Ana Rodríguez
Yakini Tchouka • Thomas Vigouroux

Supervised by Pascal Cuoq

REDOCS

TRUST IN SOFT

Context

Definitions

- X.509
- ASN.1
- Coverage
- OpenSSL - MbedTLS

Definitions

- → **X.509**
- → ASN.1
- → Coverage
- → OpenSSL - MbedTLS

X.509

You want to be able to identify yourself and know that you are talking to the right person

- Public Key Infrastructure Standard
- Certification authorities (CAs) sign certificates to certify their validity

Structure:

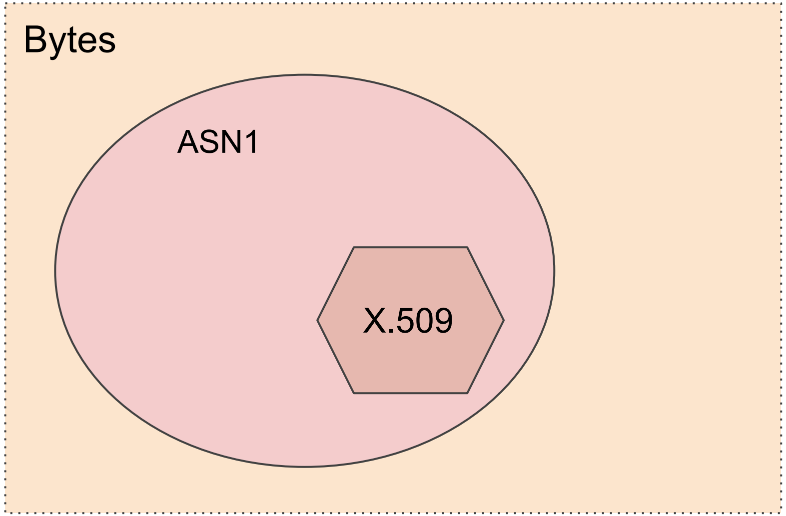
- Signature Algorithm
- Issuer: CA identification
- Validity dates
- Owner’s identity
- Key exchange algorithm and owner’s public key
- Signature Algorithm and signature Value

Definitions

- X.509
- **ASN.1**
- Coverage
- OpenSSL - MbedTLS

ASN.1

Standard interface description language

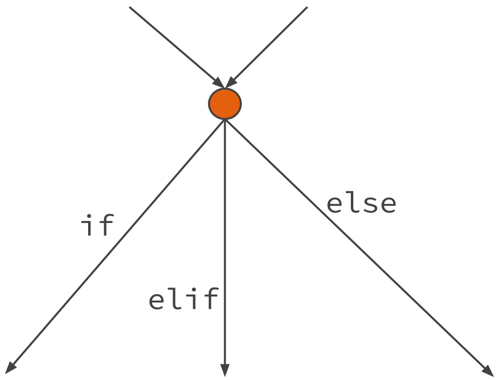


Definitions

- X.509
- ASN.1
- **Coverage**
- OpenSSL - MbedTLS

Code Coverage

Directed acyclic graph (DAG):



- Compile and test with many certificates
- Count the number of branches for each branching point reached
- Count the number or branches reached

Definitions

- ➔ X.509
- ➔ ASN.1
- ➔ Coverage
- ➔ **OpenSSL - MbedTLS**

OpenSSL

Secure communication over computer networks

Implements SSL and TLS protocols

Very convoluted implementation

Study coverage of the certificate verification

MbedTLS

Secure communication over computer networks

Implements TLS for constrained devices

Small hardware footprint

Study coverage of the certificate verification

Research Questions

Automatic generation of tests

Generating efficient tests

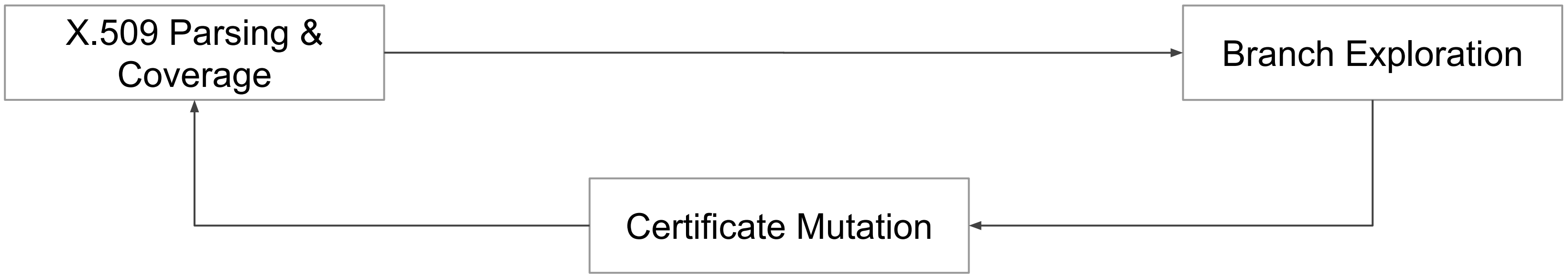
Avoid as much as possible human intelligence

Maximum coverage

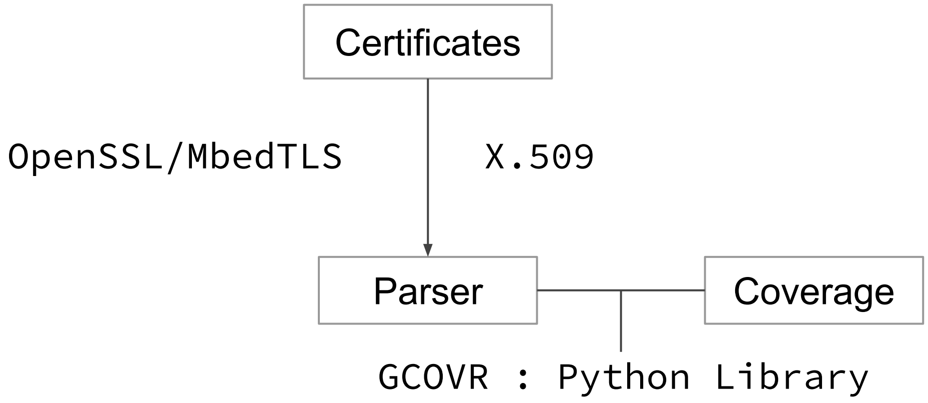
How to determine if a branch is reachable

Variables influencing a given branching point

Technical choices



X.509 Parsing & Coverage



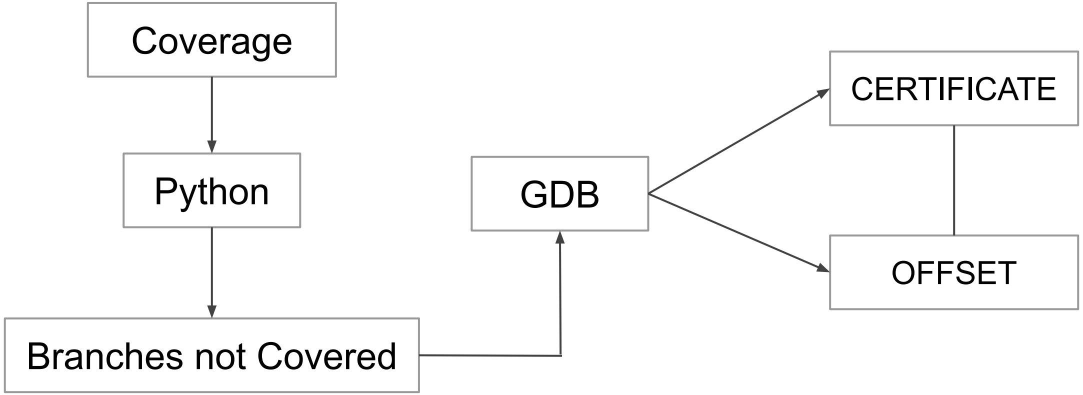
Parser:

1. OpenSSL
2. MbedTLS

Coverage:

1. GCOVr
2. LCOV

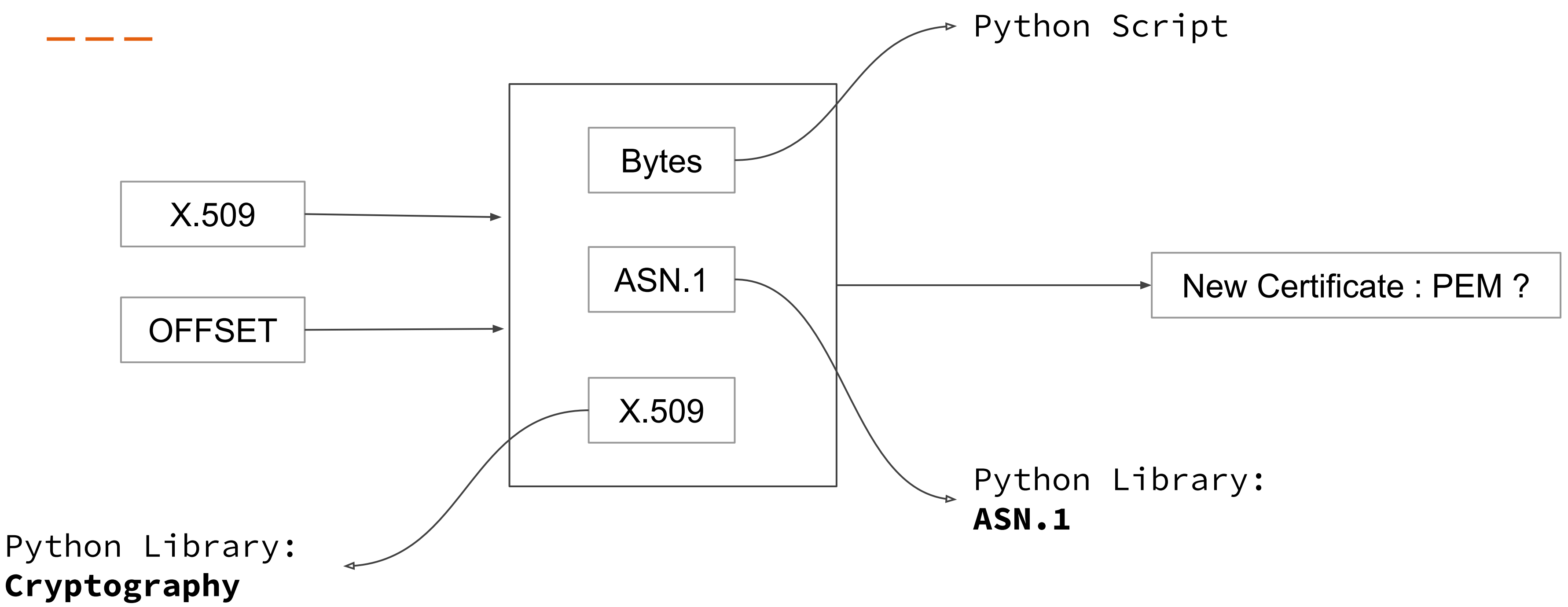
Branch Exploration



DEBUG:

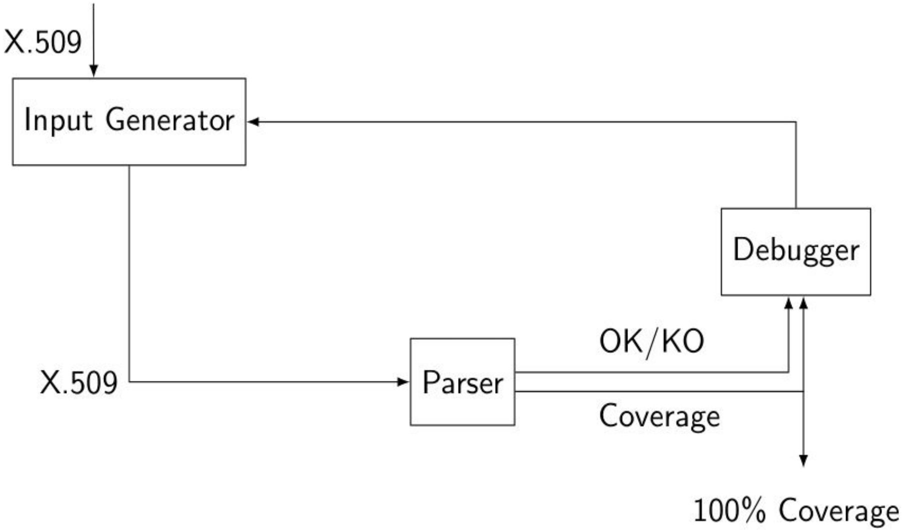
1. GDB
2. Logging in X.509 parsing script (OpenSSL/MbedTLS)

Certificate Mutation

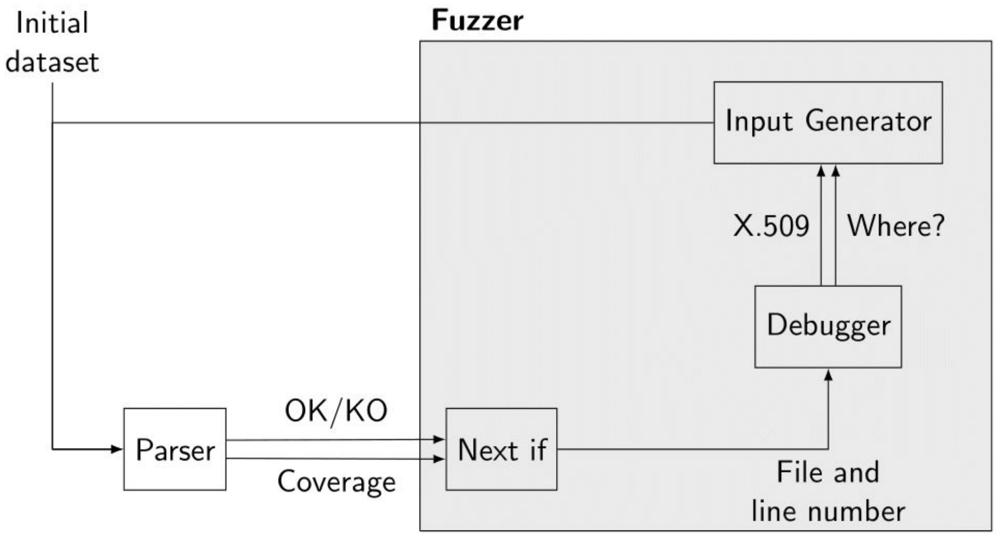


Architecture

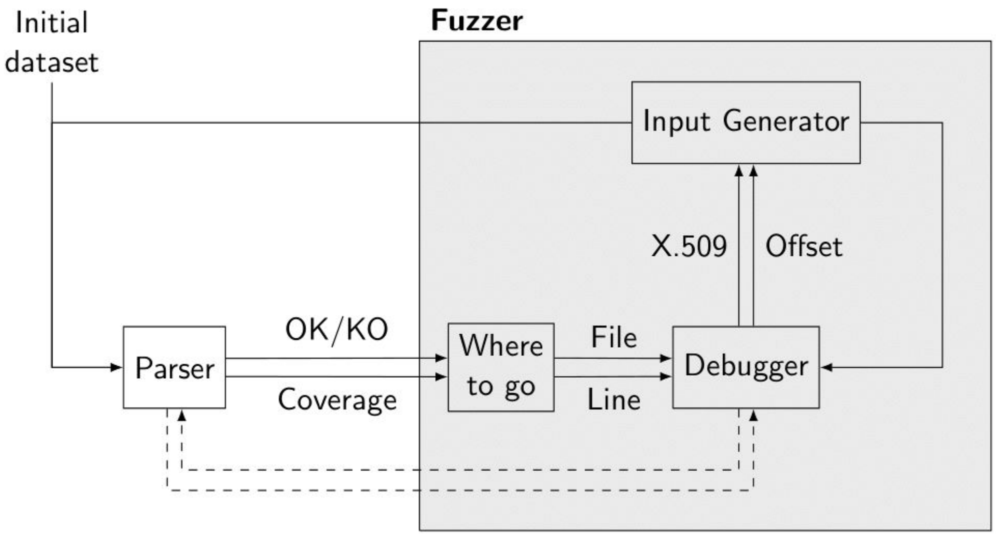
Prototype: Day 1

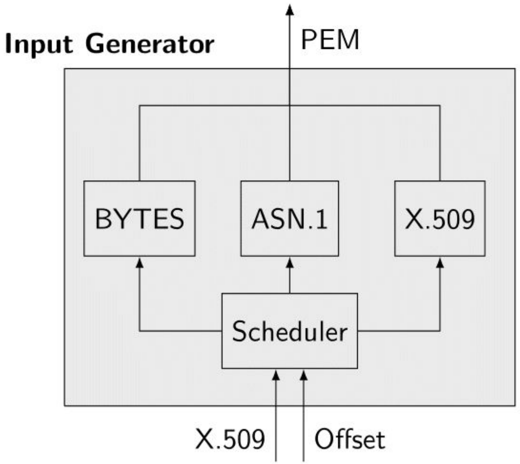


Prototype: Day 2



Prototype: Ideal Structure





PEM Generators 1

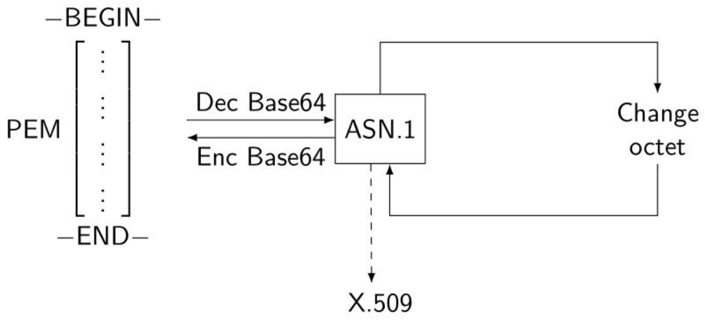


Figure: Input Generator BYTES. Invalid ASN.1 format, invalid X.509 format.

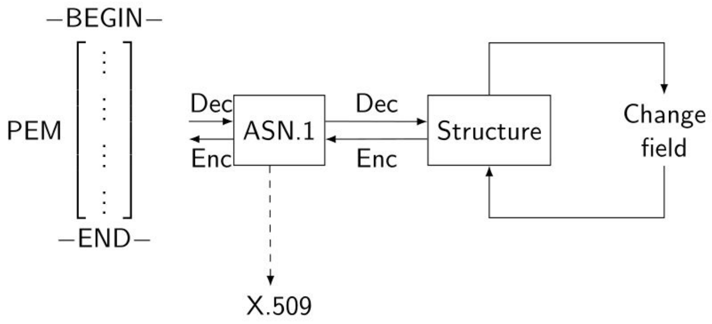


Figure: Input Generator ASN.1. Valid ASN.1 format, invalid X.509 format.

PEM Generators 2

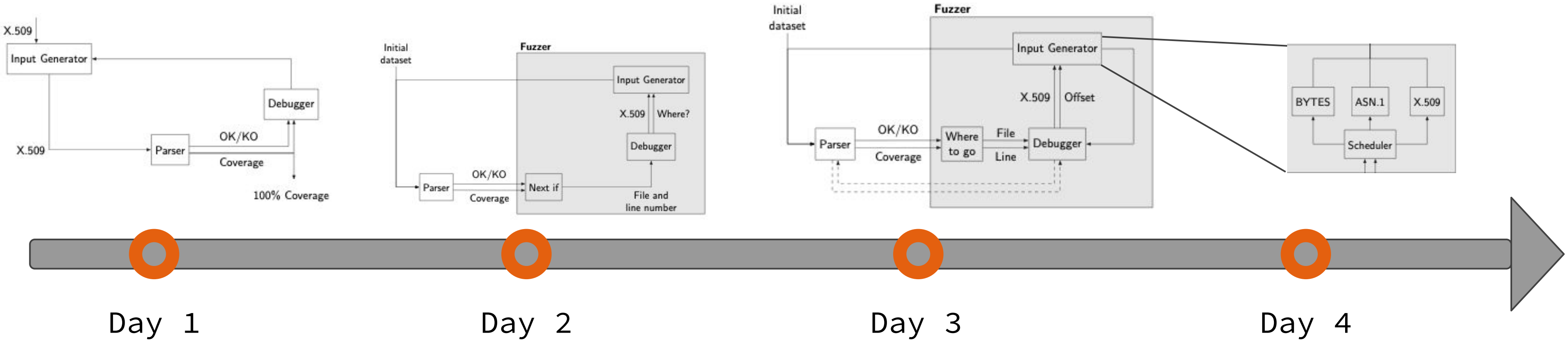
$$\text{Initial dataset} \left[\begin{array}{l} \text{Cert } 1 \\ \text{Cert } 2 \\ \text{Cert } i \left\{ \begin{array}{l} \text{Attr}^{i}\ 1 \\ \text{Attr}^{i}\ 2 \\ \text{Attr}^{i}\ 3 \\ \quad\vdots \\ \text{Attr}^{i}\ n \end{array} \right. \\ \text{Cert } n \end{array} \right.$$

$$\downarrow$$

$$\text{Cert } new \left\{ \begin{array}{c} \text{for wanted } j \in [m] \text{ sample } k \xleftarrow{\$} \text{Attribute Space} \\ \text{Attr}^{new}\ j = k \end{array} \right. .$$

Figure: Input Generator X.509. Valid X.509 format.

Results


X.509
Input Generator
Debugger
X.509
Parser
OK/KO
Coverage
100% Coverage
Initial
dataset
Fuzzer
Input Generator
X.509
Where?
Debugger
Parser
OK/KO
Next if
Coverage
File and
line number
Initial
dataset
Fuzzer
Input Generator
X.509
Offset
Parser
OK/KO
Coverage
Where
to go
File
Line
Debugger
BYTES
ASN.1
X.509
Scheduler
Day 1
Day 2
Day 3
Day 4

Day 1: Setting Up

Research questions:

- Get familiar with the subject
- Obtain a X.509 corpus
- Think about the architecture
- Install the tools

Steps:

- Verify our comprehension of the subject
- Find a corpus open licence
- Compile OpenSSL

Sticking points:

- Compile OpenSSL … with the **correct options**
- Extract the X.509 corpus
- Modify a byte in X.509 certificate

Day 2: First Prototype

Research problems:

- Extract the X.509 corpus
- Use (intelligently) the coverage tools
- Develop mutation generators

Steps:

- Bytes mutation generator
- Evolution of our architecture
- GCOVr use for coverage
- First version of a running prototype

Sticking points:

- Extract coverage info from GCOV et LCOV
- Create temporary coverage files
- Compile MbedTLS

- Extend mutations to ASN.1
- Solve bugs

Day 3: Improving our Generators

Research questions:

- Increase the mutations support by the generators
- Create temporary coverage files

Steps:

- Work on ASN.1 and X.509 mutations generator
- Prototype
- X.509 base64 structure analysis

Sticking points:

- Mutate X.509 certificates
- Automate mutations on ASN.1 structure
- Solve bugs

Day 4: Proving our Prototype ... and Making Slides ...

Research questions:

- Integration of the mutation generators
- Test prototype with a corpus of certificates
- Make good slides

Steps:

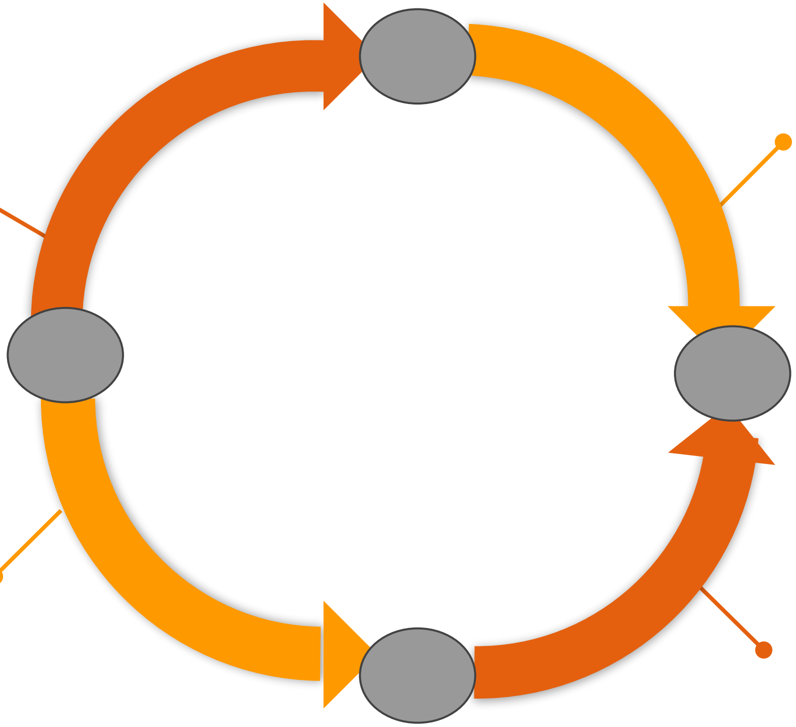
- Creation of slides
- Finishing generators
- Run prototype on OpenSSl and MbedTLS
- Optimization of the offset problem

Sticking points:

- Lookup table base64 vs. fields X.509
- Integration of the generators

Some Numbers

OpenSSL : asn1_lib.c
MbedTLS: x509_crt.c



In Short ...

- A semi-automatic prototype
- Theoretical 100% coverage
- Coverage increase of around **20%** in the best cases

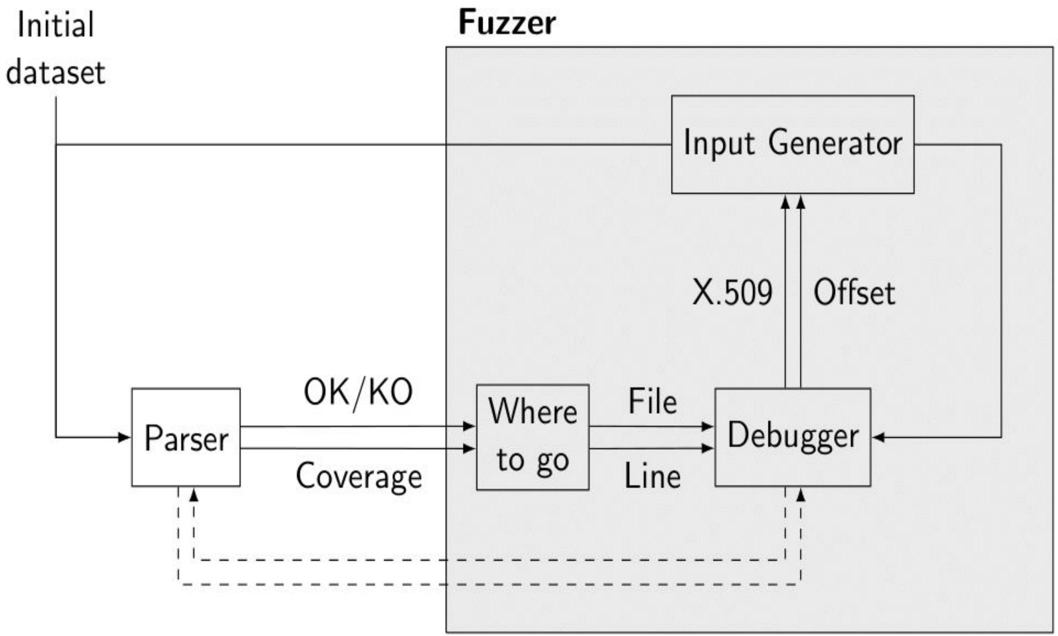
- Report on Github of 3 Undefined Behaviour(UB) on MbedTLS (with Pascal assistance) and 2 X.509 certificates from our corpus

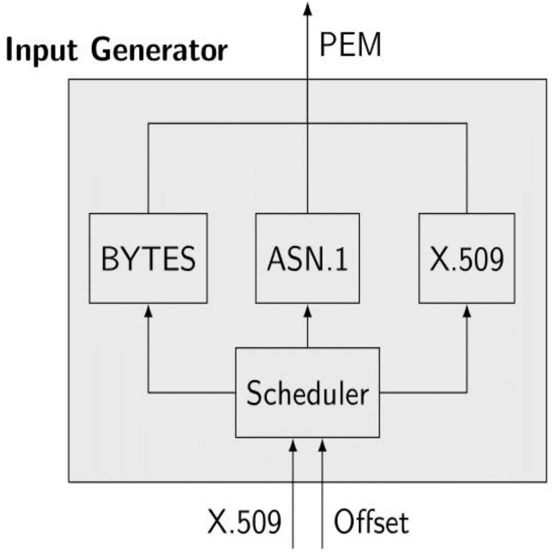
Future Works

Full automation

Enhancing mutations

Recalling the Ideal Architecture





Full Automation

Automatic offset finding

Dependency analysis (hard problem)

Heuristics

Determining mutation kind

Actually implement the scheduler

Bytes / ASN.1 / X.509 ?

How far from the current offset ?

Enhanced Mutation

Mutation scheduling

Trying promising mutations first

Mutation kind interleaving

Mutation diversity

Avoiding to try similar mutations

Test Cases Efficiency

Corpus minimization

Always generate “unique” cases

Filter out redundant cases

Focus branches of interest

Search heuristics

Metrics for projected coverage

Conclusion

A semi-automated tool

100% is feasible

State of the art:

- AFL / LibFuzzer
- Symbolic Execution / Formal methods

Thanks for your attention!

ferran.alborch@orange.com
alba.martinez-anton@lis-lab.fr
ana-margarita.rodriguez-cordero@loria.fr
yakini.tchouka@femto-st.fr
thomas.vigouroux@univ-grenoble-alpes.fr

Biblio

Python:

- Documentation lib gdb
- Documentation lib cryptography

Couverture:

- Documentation gcov, lcov et gcovr

Fuzzers

OpenSSL et MbedTLS

Demo time!

From 22 % to 56 %

Proof that “semi-automatic”-ness is a real thing