DeepFuzz: Triggering Vulnerabilities Deeply Hidden in Binaries (Extended Abstract)

Konstantin Böttinger and Claudia Eckert

DIMVA 2016

13th Conference on Detection of Intrusions and Malware & Vulnerability Assessment

7 July 2016

Donostia-San Sebastián

Agenda

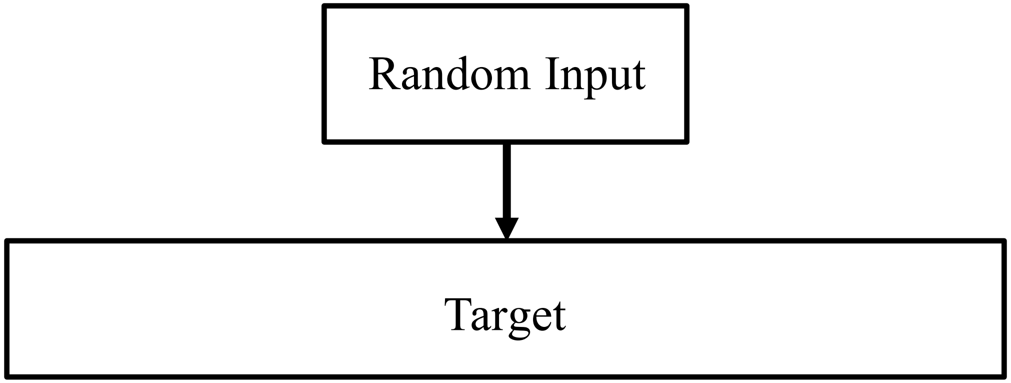
- Motivation
- Background
- Fuzzing Algorithm
- Implementation and Observations

Motivation

main goal:

improve fuzzing

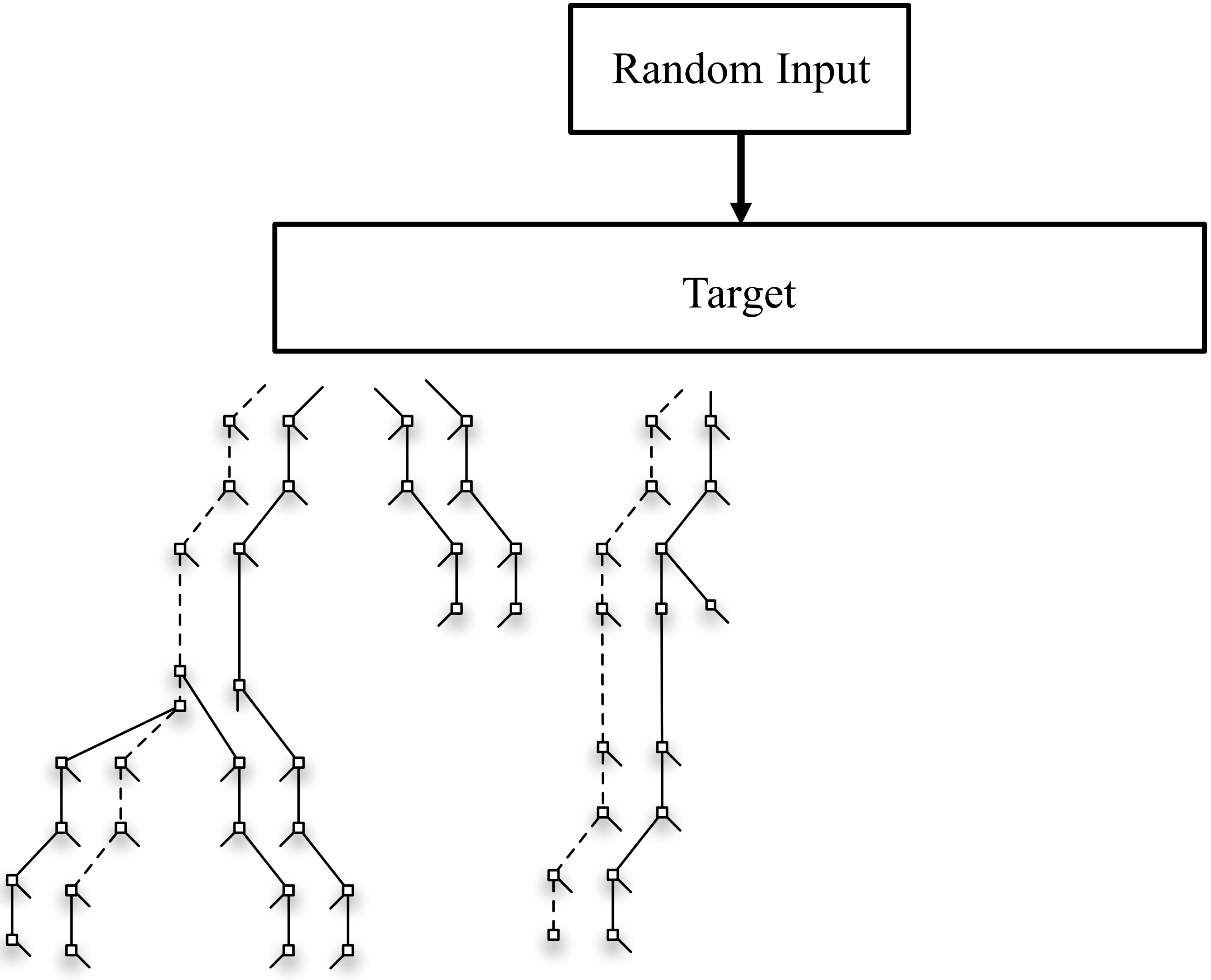
Motivation



Random Input

Target

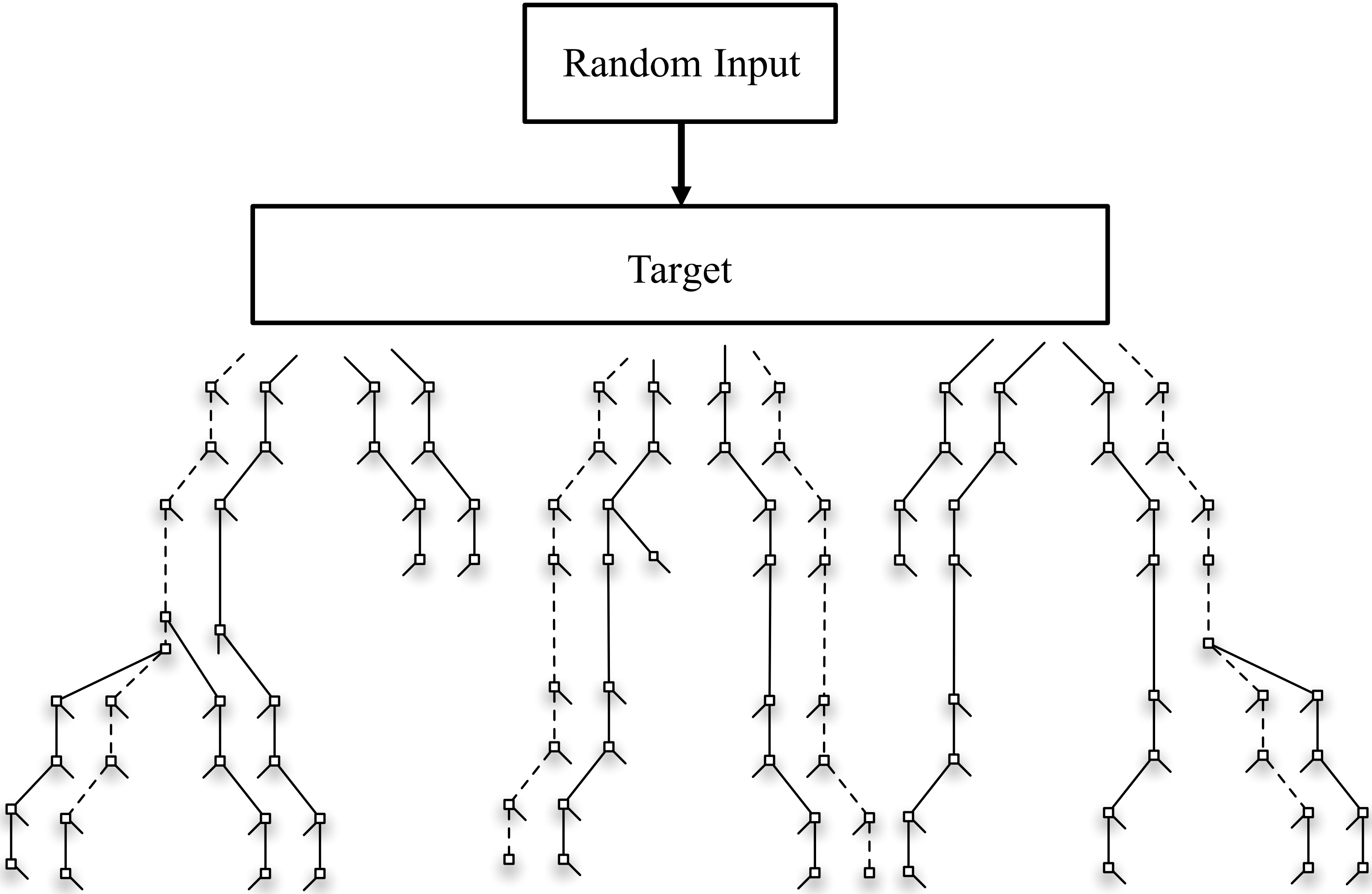
Motivation



Random Input

Target

Motivation


Random Input

Target

Motivation

```
#include <stdint.h>
...
int check( uint64_t num ){
        if( num == UINT64_C(0) )
                assert( false );
}
```

- probability of 2^{-64} to pass the `if` statement
  - → *fuzzing wall*

Motivation

How can we fuzz through fuzzing walls

to reach deep layers of the program?

Background

- Concrete symbolic (concolic) execution
  - assign symbolic representations to input variables of a program and generate formulas over the symbols according to the transformations in the program execution
  - program is initially executed with arbitrary concrete input values and symbolic constraints over the symbols are generated along the program execution path
  - one of the collected branch conditions is negated and together with the remaining constraints given to an SMT solver
  - The solution (*model*) generated by the SMT solver is injected as new input into the program, which now takes the branch alternative when executed
  - effective for complex arithmetic operations, pointer manipulations, calls to external library functions, or system calls

Background

however

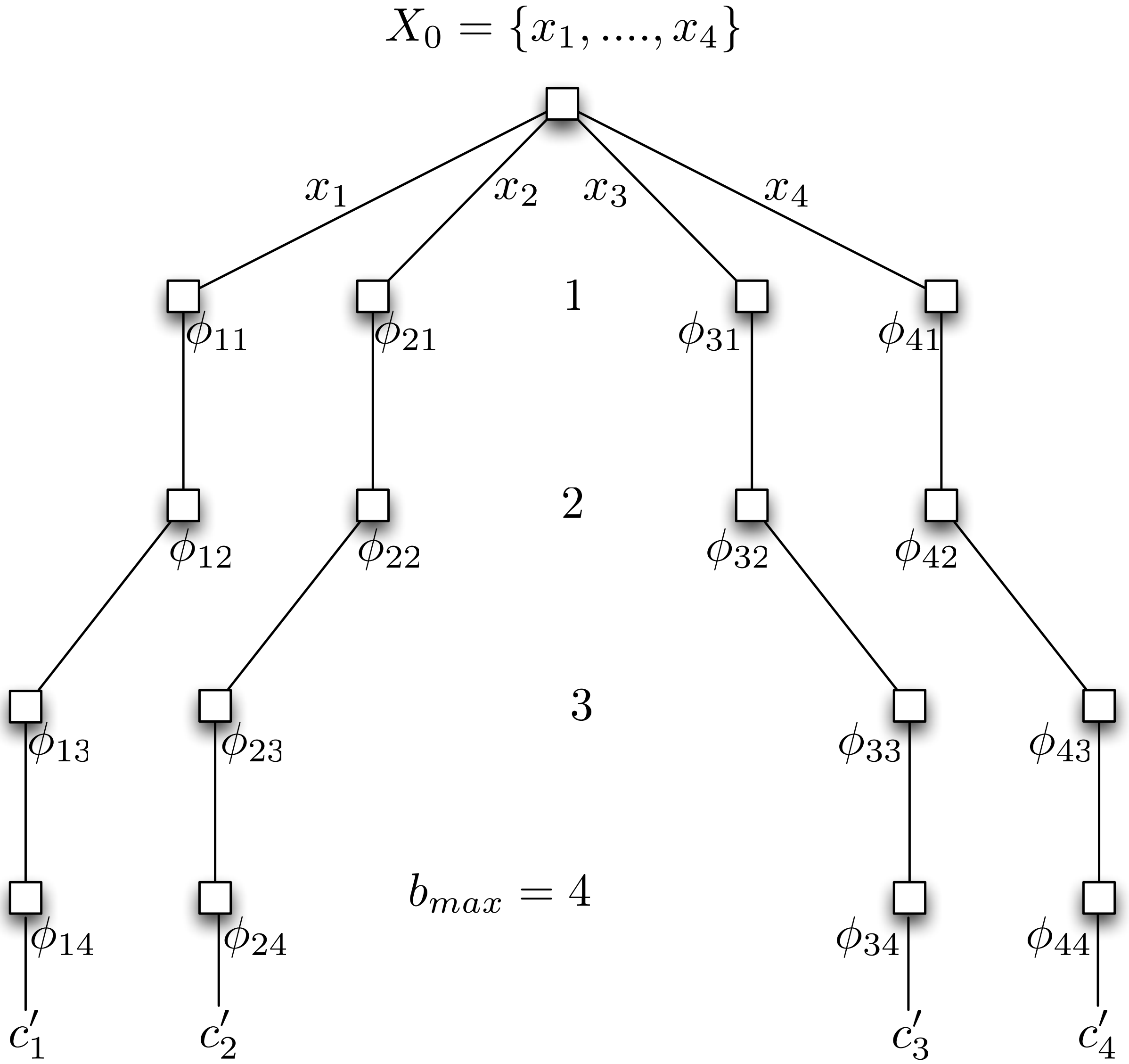
path explosion

Background

- want to fuzz deep areas of a program
  - find a way to construct execution paths into such areas
  - delay path explosion until we have found such a tunnel

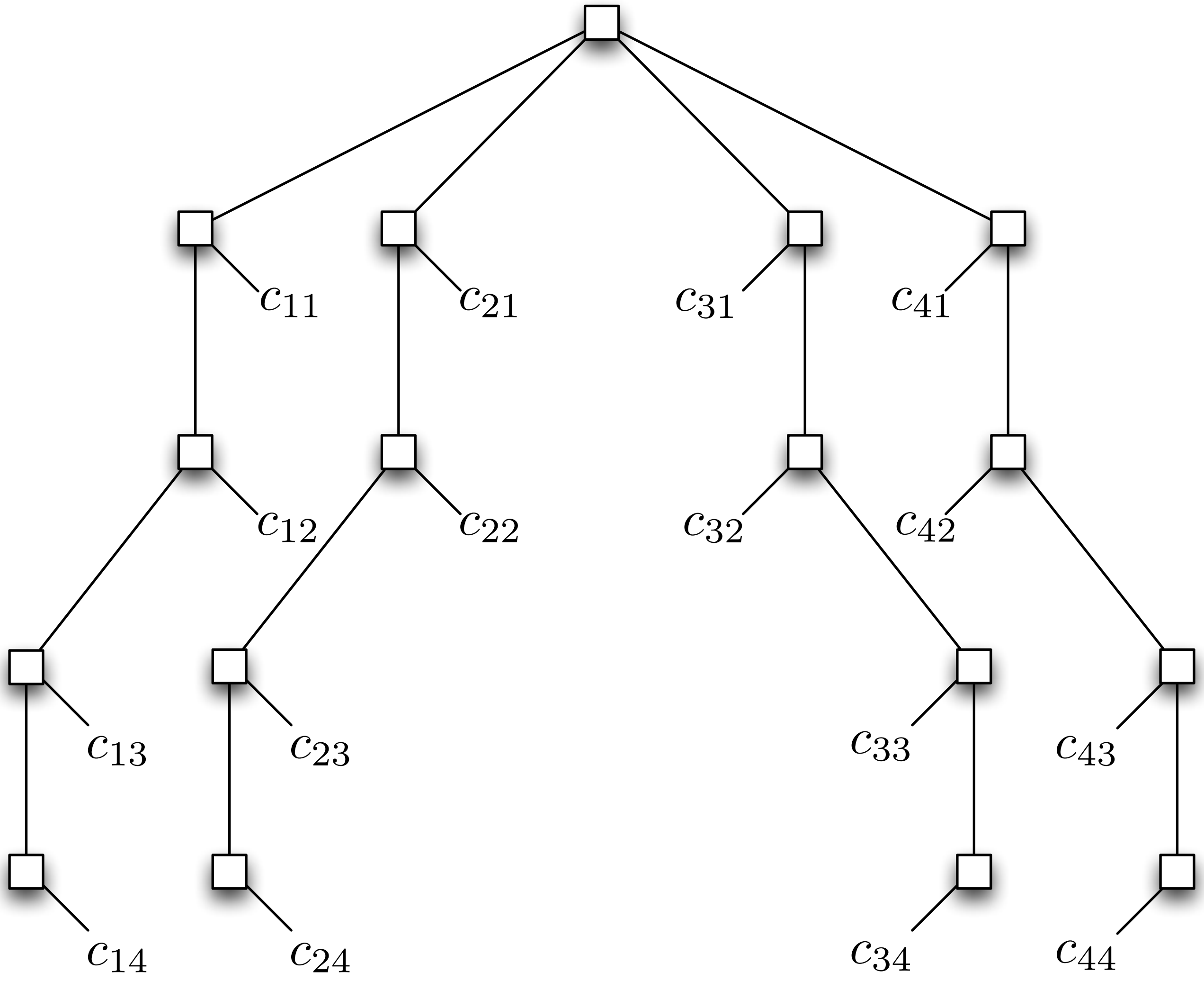
- idea:
  - interleave concolic execution with constrained fuzzing
  - assign weights (corresponding to fuzzing performance) to the explored paths after each concolic execution step in order to select the ones with highest probability

Algorithm


$X_0 = \{x_1, ..., x_4\}$
x_1
x_2
x_3
x_4
1
ϕ_{11}
ϕ_{21}
ϕ_{31}
ϕ_{41}
2
ϕ_{12}
ϕ_{22}
ϕ_{32}
ϕ_{42}
3
ϕ_{13}
ϕ_{23}
ϕ_{33}
ϕ_{43}
$b_{max} = 4$
ϕ_{14}
ϕ_{24}
ϕ_{34}
ϕ_{44}
c_1'
c_2'
c_3'
c_4'

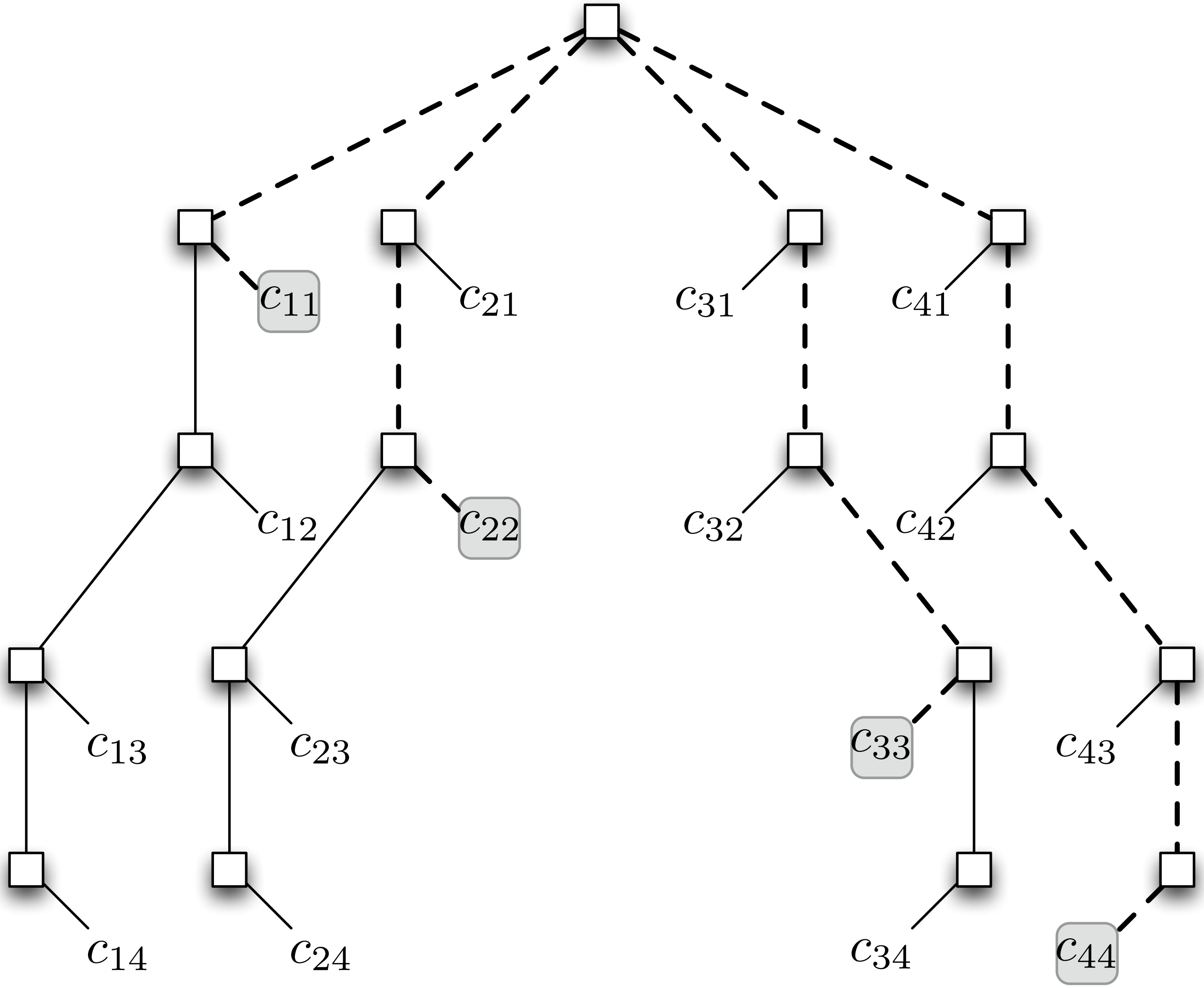
Algorithm

$$\{x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{24}, x_{31}, x_{32}, x_{33}, x_{34}, x_{41}, x_{42}, x_{43}, x_{44}\}$$



Algorithm

$$\{\boxed{x_{11}}, x_{12}, x_{13}, x_{14}, x_{21}, \boxed{x_{22}}, x_{23}, x_{24}, x_{31}, x_{32}, \boxed{x_{33}}, x_{34}, x_{41}, x_{42}, x_{43}, \boxed{x_{44}}\}$$



Algorithm

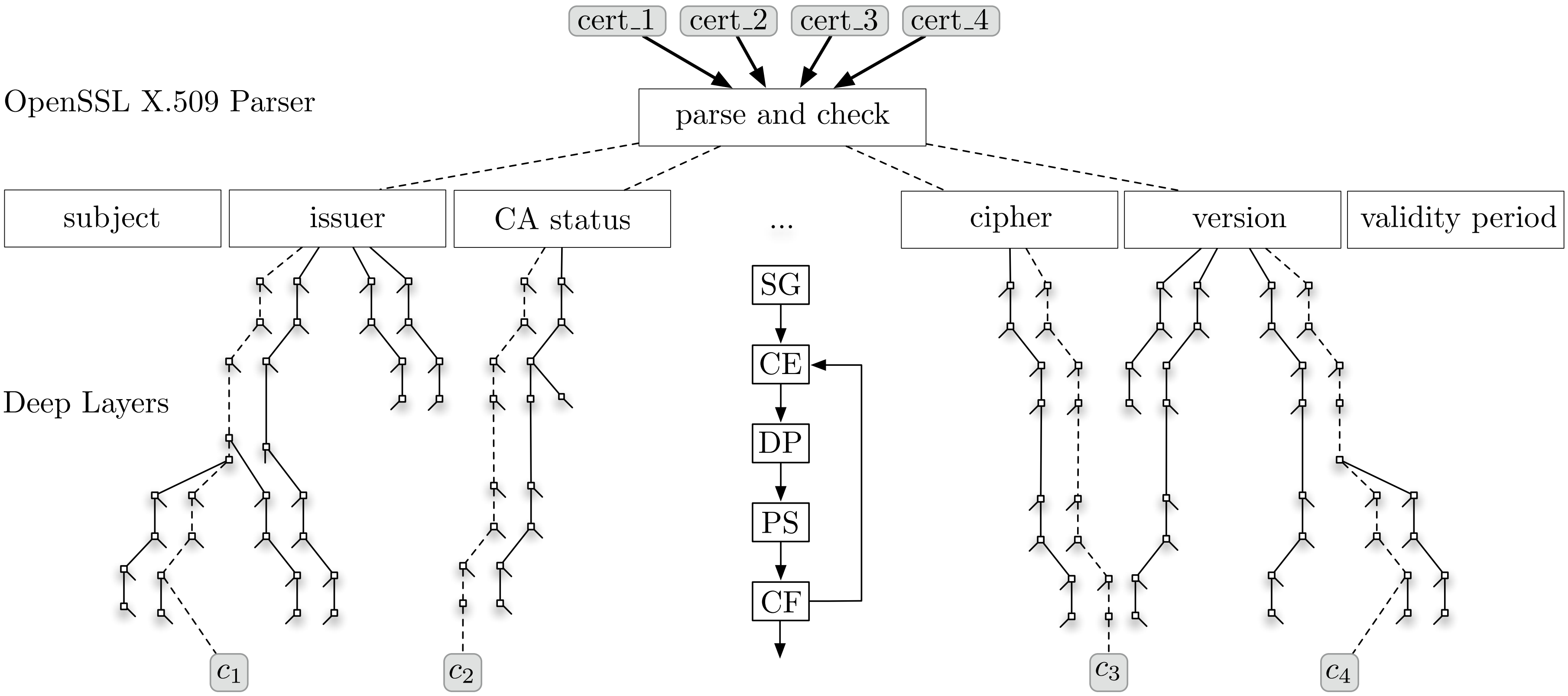
```
**Input:** Program $P$, Parameters $m,\ k_{min},\ T_0,\ T_1,\ T_2,\ b_{max}$

$X_{seed} \leftarrow \mathrm{SG}\ (P)$
do:
    $\Phi = \emptyset$
    $C = \emptyset$
    for each $x$ in $X_{seed}$ do:
        $c,\ \phi\ \leftarrow \mathrm{CE}\ (x, b_{max})$
        append $\phi$ to $\Phi$
        append $c$ to $C$
    $Prob\ \leftarrow\ \mathrm{DP}\ (\Phi, C, T_0)$
    $C_{high}\ \leftarrow\ \mathrm{PS}\ (Prob, C)$
    $X_{seed}\ \leftarrow\ \mathrm{CF}\ (C_{high}, \Phi, T_1)$

while $\neg$ condition (11)    ; i.e. $\left(\sum_{i=1}^{m} k_i(\phi_i, T_0) \geq k_{min}\right)$

$\mathrm{CF}(C_{high}, \Phi, T_2)$
```

Implementation and Observations



OpenSSL X.509 Parser

cert_1 cert_2 cert_3 cert_4

parse and check

subject

issuer

CA status

...

cipher

version

validity period

Deep Layers

SG

CE

DP

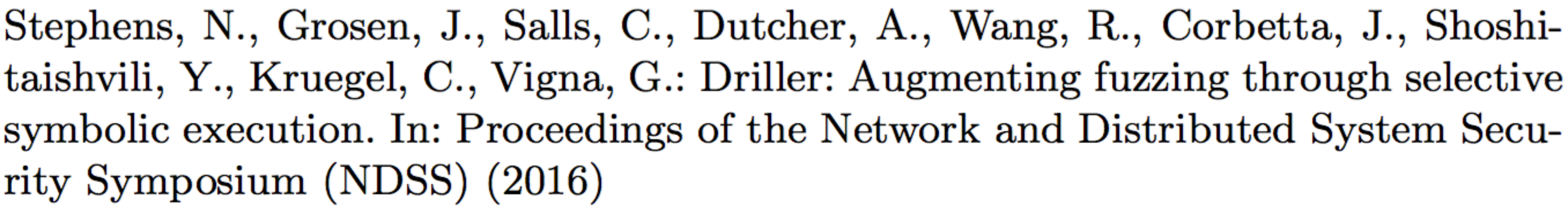
PS

CF

c_1 c_2 c_3 c_4

Related Work

Driller



Stephens, N., Grosen, J., Salls, C., Dutcher, A., Wang, R., Corbetta, J., Shoshitaishvili, Y., Kruegel, C., Vigna, G.: Driller: Augmenting fuzzing through selective symbolic execution. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2016)

Summary and Conclusion

- Proposal of a new search heuristic that delays path explosion effectively into deeper layers of the tested binary
- Novel technique to assign probabilities to execution paths
- Algorithm combining initial seed generation, concolic execution, distribution of path probabilities, path selection, and constrained fuzzing

Thank you for your attention!

Konstantin Böttinger

Product Protection and Industrial Security

Fraunhofer Institute for Applied and Integrated Security (AISEC)

Phone: +49 89 3229986-163

E-Mail: konstantin.boettinger@aisec.fraunhofer.de

Internet: www.aisec.fraunhofer.de