Fuzzing Methods

Offensive Security

Moritz Rupp

Hochschule Albstadt-Sigmaringen

SS 22

Contents

Fuzzing background

- Development produces bugs, errors and unintentionally behaviour
  $\rightarrow$ Gateway for vulnabilities and exploits
- Software testing tries to oppose that
- Many different approaches exist $\rightarrow$

Software testing

Manual testing

- Code reviews, manuel search for vulnabilities
- Time consuming, expensive

Static analysis

- Automatically examine source code before the programm is run
- Pattern analysis
- Control flow graph, data flow analysis
- Expensive tooling

Dynamic analysis

- Automatically examine a programm while its's been run
- Execute and input data in real-time

Fuzz-testing

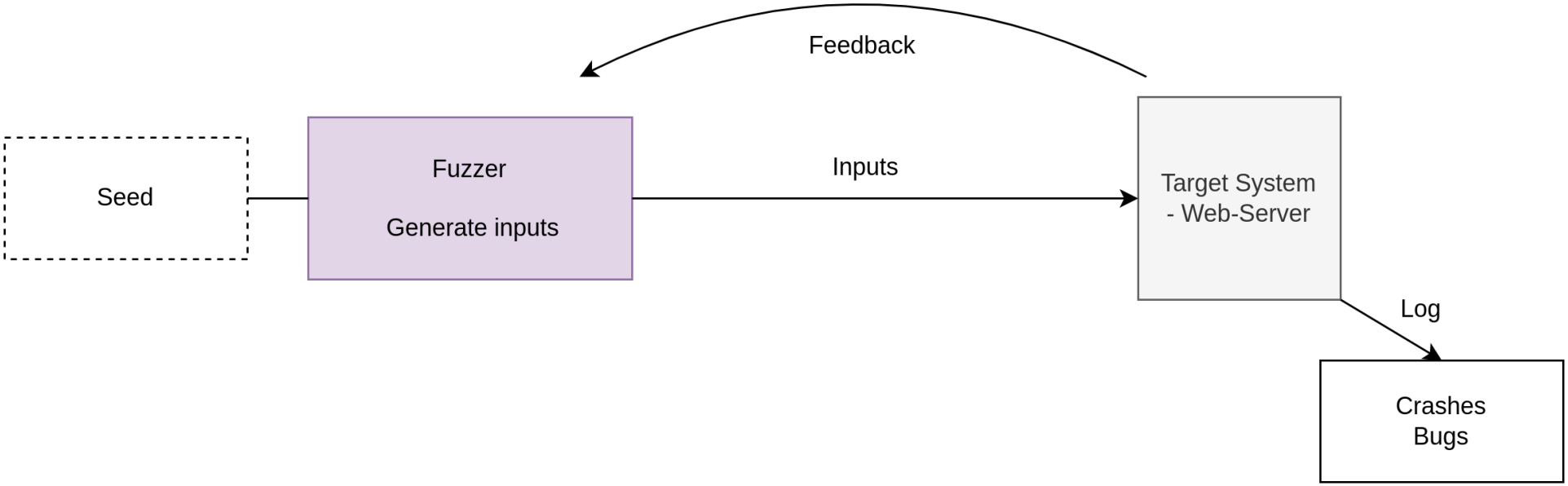
$\Rightarrow$ Fuzz-testing is the cutting edge of dynamic analysis

- Input forget data in real-time
- Monitor the system behaviour
- Listen for exceptions
- Provide feedback

Functionality

1. Identify target interfaces
   $\rightarrow$ portscanning, code reviews
2. Generate inputs
   $\rightarrow$ mutation, generation based fuzzing
3. Feed these inputs to the target system
4. Monitor for exceptions
5. Log exceptions

Basic Fuzzing application



[1]

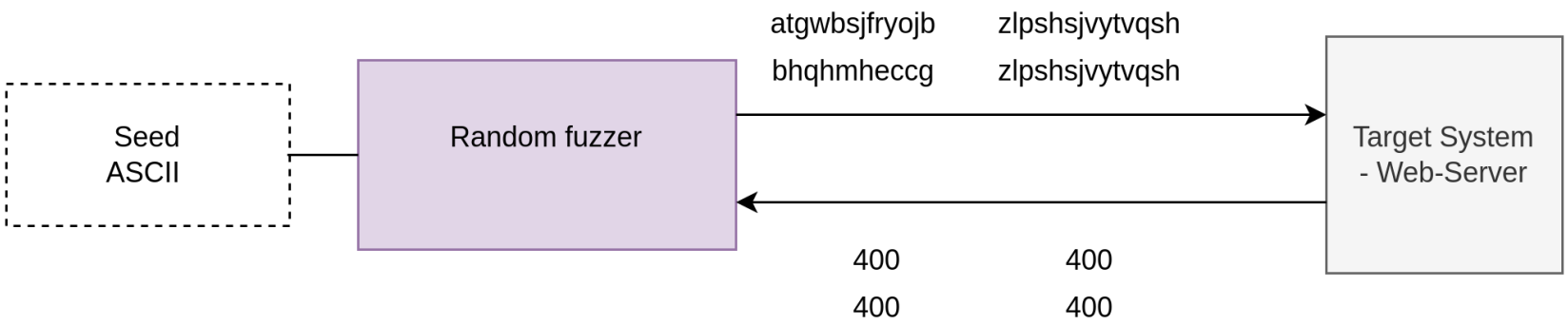
Fuzzing Methods

- Huge landscape of applications and infrastructers
  $\rightarrow$ Web-applications, networks, binaries etc.
  $\Rightarrow$ No generall solution
- Different targets expect different inputs
- How do we generate those inputs?
  $\Rightarrow$ Random fuzzing
  $\Rightarrow$ Mutation based fuzzing
  $\Rightarrow$ Generation based fuzzing

Random Fuzzing

- Earliest fuzzing approach
- Generate pseudo random values of a given seed
- Mostly usefull for black box testing
  $\rightarrow$ Closed source projects
- Will generate lots of rejected test-cases

Random based fuzzer



[2]

Mutation based Fuzzing

- Generate test-cases based on already existing data
  $\Rightarrow$ Record valid inputs
  $\Rightarrow$ Mutate these inputs
- Randomly or after fixed patterns
- No need for deep protocol knowledge
- Flip random nummber of bits

Mutation based Fuzzer

```
seed = "https://www.hs-albsig.de/suchergebnisse?tx_solr%5Bq%5D=search"
mutation_fuzzer = MutationFuzzer(seed=[seed])
[mutation_fuzzer.fuzz() for i in range(5)]
['https://www.hs-albsig.de/suchergebnisse?tx_solr%5Bq%5D=searchf'
'hLtSs://kww.hR-aObsig.dz/PusSergebsdishje?tx_slölrerBq%5D!ReaTR'
'Tt?ps:s/wiw.hssdaalbsiw.de/sRchergebEWisseWTgtx_solGD%5Bf%5Ds=s'
'hSt?sR//wwrphs-albsUgwdE/such-ugeEGsbnisse?tRsolr%5Bq%5D=search'
'https://www.hL-albsig.de/sasd?-ergebniRe?tx_ewlrew5Bq%5DewsRarch'
'https://www.hs-albsig.de/suchergebnisse?tas_soasdr%5Bd5D=LeaESch']
```

[3]

- Determine the seed correctly
  $\Rightarrow$ 'suchergebnisse?fuzz'
- Setup fixed grammars
  $\Rightarrow$ http/ fuzz

Generation based Fuzzing

- Generate inputs from scratch
- Based on the specification and format
- Protocol knowledge is importand
- Generated inputs are semi-valid
- Will barely generate rejected test-cases
- Have to be developed from scratch for every protocol/application

- Assemple every part of an input sepperatly
- Assign rules to these indivual parts

Creation rules in Sully

s_static	set a static value
s_string	fuzz the provided string
s_delim	fuzz delimters

Fuzzing of an http request

→ GET /index.html HTTP ...

```
s_static('GET')
s_delim(' ')
s_delim('/')
s_string('index)
s_static('.')
s_string('html')
s_delim(' ')
s_string('HTTP')
```

Tooling

- Fuzzing is almost always executed with tooling
- Manuel testing only usefull in rare cases
- Many different tools exist for different fuzzing methods/targets
  $\Rightarrow$ OSS-Fuzz, Wfuzz, Sully, FuzzDB ...

AFL

$\Rightarrow$ American fuzzing loop

- Open-source
- Mutation and generation based Fuzzer
- Widely used across many target systems
- Responsible for lots of relevant findings
  $\rightarrow$ X.Org Server, PHP, OpenSSL, Firefox, Bash ...

Practical example

Conclusion

- Fuzzing effectively locates bugs
  $\rightarrow$ Mostly simple bugs
- Importand to choose the right method for the corresponding target
- Boosts stability and security of systems
- Quite complex to implement
- Growing usage
  $\rightarrow$ in combination with maschine learning etc.

Quellen

[1] H. Liang, X. Pei, X. Jia, W. Shen and J. Zhang, "Fuzzing: State of the Art," in IEEE Transactions on Reliability, vol. 67, no. 3, pp. 1199-1218, Sept. 2018, doi: 10.1109/TR.2018.2834476.
[2] Li, J., Zhao, B. & Zhang, C. Fuzzing: a survey. Cybersecur 1, 6 (2018). https://doi.org/10.1186/s42400-018-0002y
https://www.ionos.com/digitalguide/websites/web-development/what-is-fuzzing/
https://owasp.org/www-community/Fuzzing
Grammar-based whitebox fuzzing Authors Patrice Godefroid Adam Kiezun Michael Y. Levin Authors Info & Claims