

Fuzzing Methods

Offensive Security

Moritz Rupp

Hochschule Albstadt-Sigmaringen

SS 22

Contents

- 1 Fuzzing background
 - Software testing
 - Fuzz-testing
- 2 Functionality
- 3 Fuzzing Methods
 - Random fuzzing
 - Mutation based fuzzing
 - Generation based fuzzing
- 4 Tooling
- 5 Practical example
- 6 Conclusion

Fuzzing background

- Development produces bugs, errors and unintentionally behaviour
→ Gateway for vulnerabilities and exploits
- Software testing tries to oppose that
- Many different approaches exist

Software testing

Manuel testing

- Code reviews, manuel search for vulnerabilities
- 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

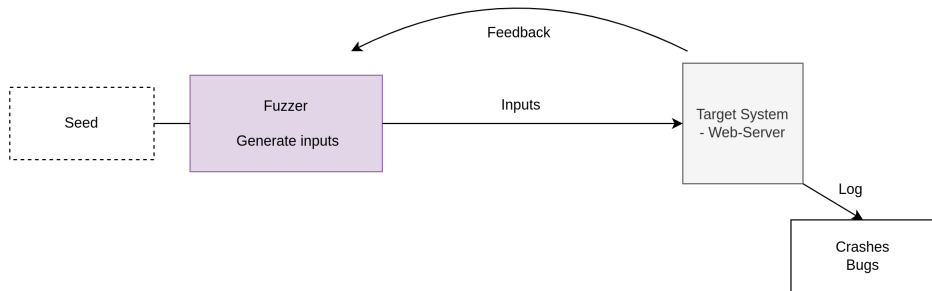
⇒ Fuzz-testing is the cutting edge of dynamic analysis

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

Functionality

- ➊ Identify target interfaces
→ portscanning, code reviews
- ➋ Generate inputs
→ mutation, generation based fuzzing
- ➌ Feed these inputs to the target system
- ➍ Monitor for exceptions
- ➎ Log exceptions

Basic Fuzzing application



[1]

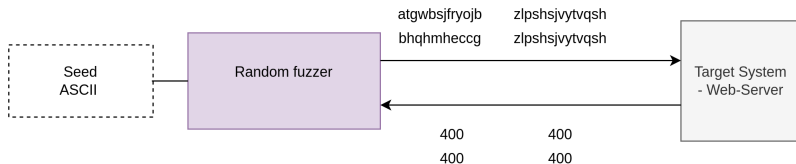
Fuzzing Methods

- Huge landscape of applications and infrastructures
 - Web-applications, networks, binaries etc.
 - ⇒ No general solution
- Different targets expect different inputs
- How do we generate those inputs?
 - ⇒ Random fuzzing
 - ⇒ Mutation based fuzzing
 - ⇒ Generation based fuzzing

Random Fuzzing

- Earliest fuzzing approach
- Generate pseudo random values of a given seed
- Mostly usefull for black box testing
 - 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
 - ⇒ Record valid inputs
 - ⇒ Mutate these inputs
- Randomly or after fixed patterns
- No need for deep protocol knowledge
- Flip random number of bits

Mutation based Fuzzer

```
1 seed = "https://www.hs-albsig.de/suchergebnisse?tx_solr%5Bq%5D=search"
2 mutation_fuzzer = MutationFuzzer(seed=[seed])
3 [mutation_fuzzer.fuzz() for i in range(5)]
4 ['https://www.hs-albsig.de/suchergebnisse?tx_solr%5Bq%5D=searchf'
5 'hLtSs://kww.hR-a0bsig.dz/PusSergebsdishje?tx_slölrerBq%5D!ReaTR'
6 'Tt?ps:s/wiw.hssdaalbsiw.de/sRchergebEWisseWTgtx_solGD%5Bf%5Ds=s'
7 'hSt?sR//wwrphs-albsUgwdE/such-ugeEGsbnisse?tRsolr%5Bq%5D=search'
8 'https://www.hL-albsig.de/sasd?-ergebniRe?tx_ewlrew5Bq%5DewsRarch'
9 'https://www.hs-albsig.de/suchergebnisse?tas_soasdr%5Bd%5D=LeaESch']
```

- Determine the seed correctly
⇒ 'suchergebnisse?fuzz'
- Setup fixed grammars
⇒ http/ fuzz

Generation based Fuzzing

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

- Assemble 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 delimiters

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
- Manual testing only usefull in rare cases
- Many different tools exist for different fuzzing methods/targets
⇒ OSS-Fuzz, Wfuzz, Sully, FuzzDB ...

⇒ American fuzzing loop

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

Practical example

Conclusion

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

[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