Fuzzing

Eine Einführung in die Welt des Fuzzings

Sebastian Peschke

13.12.2024

Hochschule für angewandte Wissenschaften Hof

About Me

Name: Sebastian Peschke
Ausbildung: B.Sc. Mobile Computing (Hochschule für angewandte Wissenschaften Hof)
Kontakt:
sebastian.peschke@hof-university.de
GitHub:
https://github.com/ItsMagick

Einführung

Was ist Fuzzing?

"Fuzzing is a vulnerability discovery solution that resonates with random-mutation, feedback-driven, coverage-guided, constraint-guided, seed-scheduling, and target-oriented strategies. Each technique is wrapped beneath the black-, white-, and grey-box fuzzers to uncover diverse vulnerabilities."Sanoop et al. [1]

Was ist Fuzzing?

"Fuzzing is a vulnerability discovery solution that resonates with random-mutation, feedback-driven, coverage-guided, constraint-guided, seed-scheduling, and target-oriented strategies. Each technique is wrapped beneath the black-, white-, and grey-box fuzzers to uncover diverse vulnerabilities."Sanoop et al. [1]

Was ist Fuzzing?

- Fuzzing ist eine Methode zur Entdeckung von Schwachstellen in Software
- Es gibt verschiedene Arten von Fuzzing-Ansätzen
- Diese Ansätze werden mit verschiedenen Strategien kombiniert

Fuzzing Ansätze

- Black-Box Fuzzing: Keine Kenntnis über den Quellcode
- White-Box Fuzzing: Kenntnis über den Quellcode
- Grey-Box Fuzzing: Teilweise Kenntnis über den Quellcode

Wieso Fuzzing?

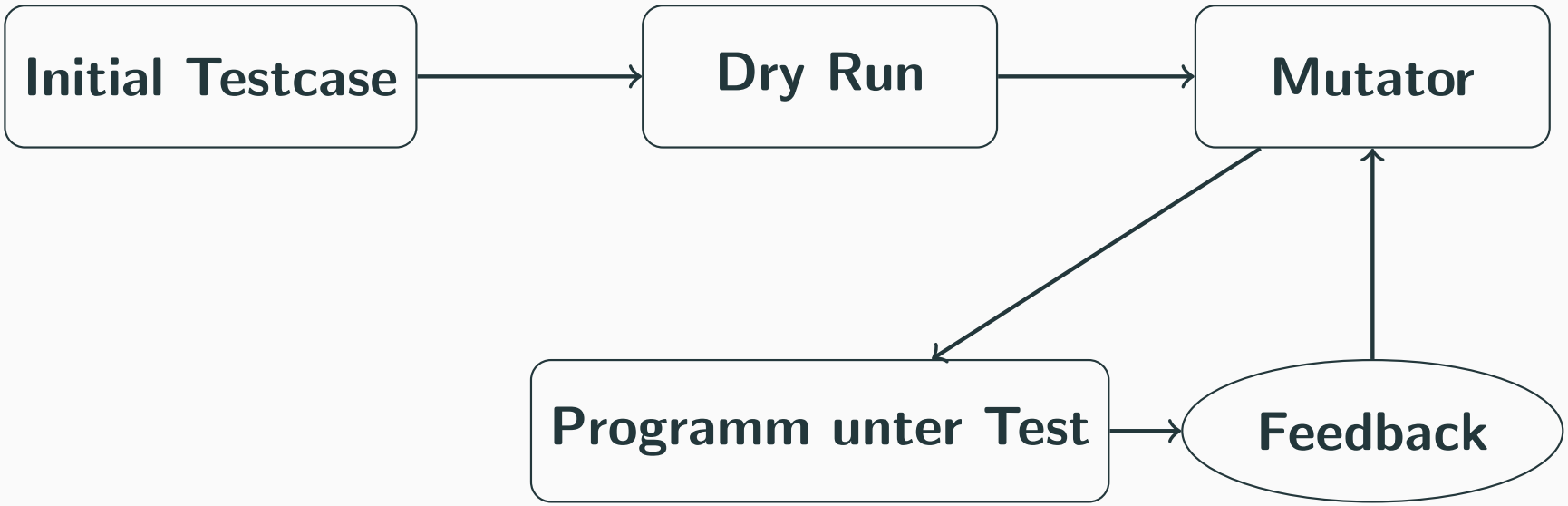
Fuzzing ist eine:

- Effektive Methode zur Entdeckung von Schwachstellen
- Automatisierte Methode
- Schnelle(-re?) Methode
- Kostengünstige Methode

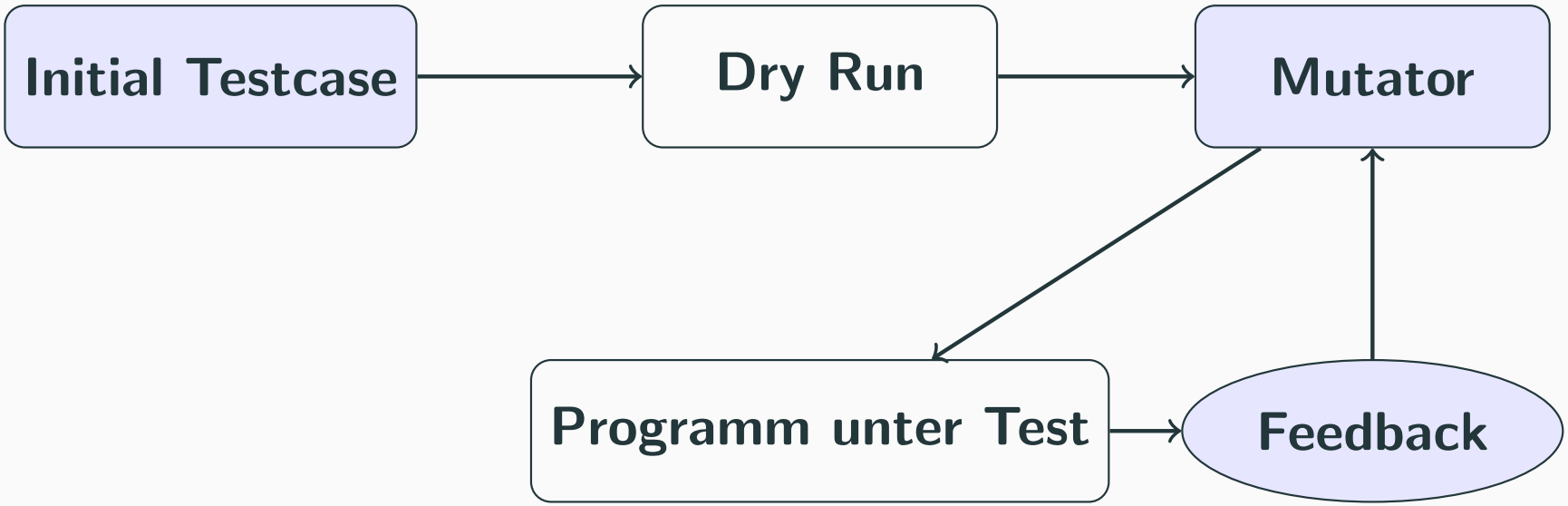
Fuzzing Anwendungsgebiete

- Binary Fuzzing
- Betriebssystem Fuzzing
- Netzwerkprotokoll Fuzzing
- Firmware Fuzzing
- IoT/Embedded Fuzzing
- ...

Wie Funktionert Fuzzing?



Wie Funktionert Fuzzing?



Demo

Herausforderung von Fuzzing

- Lückenhafte Codeabdeckung
- Unzureichende Testfälle
- Verstehen von komplexen Codepfaden
- Nichtdeterministische Programme und zustandsabhängige Systeme

Das Innere eines Fuzzers

Was steckt in einem Fuzzer?

- Corpus
- Input Generator
- Mutator
- Laufzeitumgebung für das zu untersuchende System
- Monitoring
- Strategie

Corpus

- Sammlung von Testcases (initial händisch)
- Wird durch den Fuzzer verwaltet
- Wird durch den Mutator verändert
- Wird durch den Input Generator erweitert/verkleinert

Input Generator

Am Beispiel von AFL(American Fuzzy Lop):

- Generiert Testcases anhand des Corpus
- Verwendet bevorzugt Testcases, die zu einer höheren Codeabdeckung führen

Mutator

- Verändert Testcases
- Mutationsstrategien:
  - Bitflips
  - Byteflips
  - Arithmetische Operationen
  - Block Operationen
  - Splicing
  - ...

Laufzeitumgebung

- Stellt das zu untersuchende System bereit
- Am weitesten verbreitet: QEMU
- Kann auf verschiedene Arten konfiguriert werden
- Sorgt für Isolation des zu untersuchenden Systems
- Ermöglicht Cross-Plattform Kompatibilität (z.B. Fuzzing von ARM auf x86 Host)

Einschub: Low Level Fundamentals

- Was ist ein Basic Block?
- Was ist Codeabdeckung?
- Was ist ein Codepfad?

Basic Block: Ein Basic Block ist eine Sequenz von Anweisungen, die von einem Punkt im Programmfluss bis zu einem Sprungbefehl führen.
Codeabdeckung: Codeabdeckung ist ein Maß dafür, wie viele Basic Blocks eines Programms durch Testcases erreicht werden.
Codepfad: Ein Codepfad ist eine Sequenz von Basic Blocks, die durch einen Testcase erreicht werden.

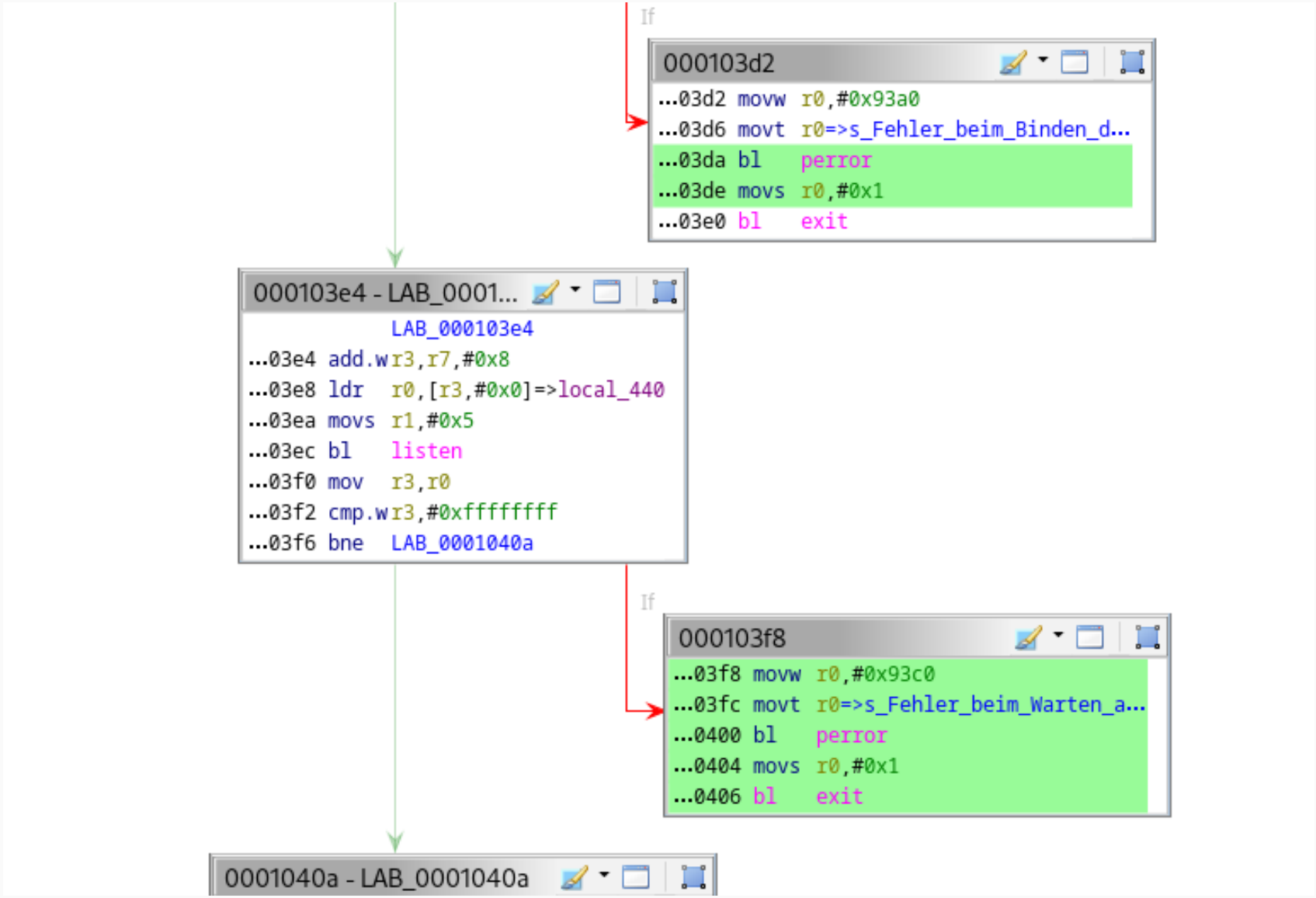


Figure 1: Basic Blocks am Beispiel eines selbst implementierten TCP

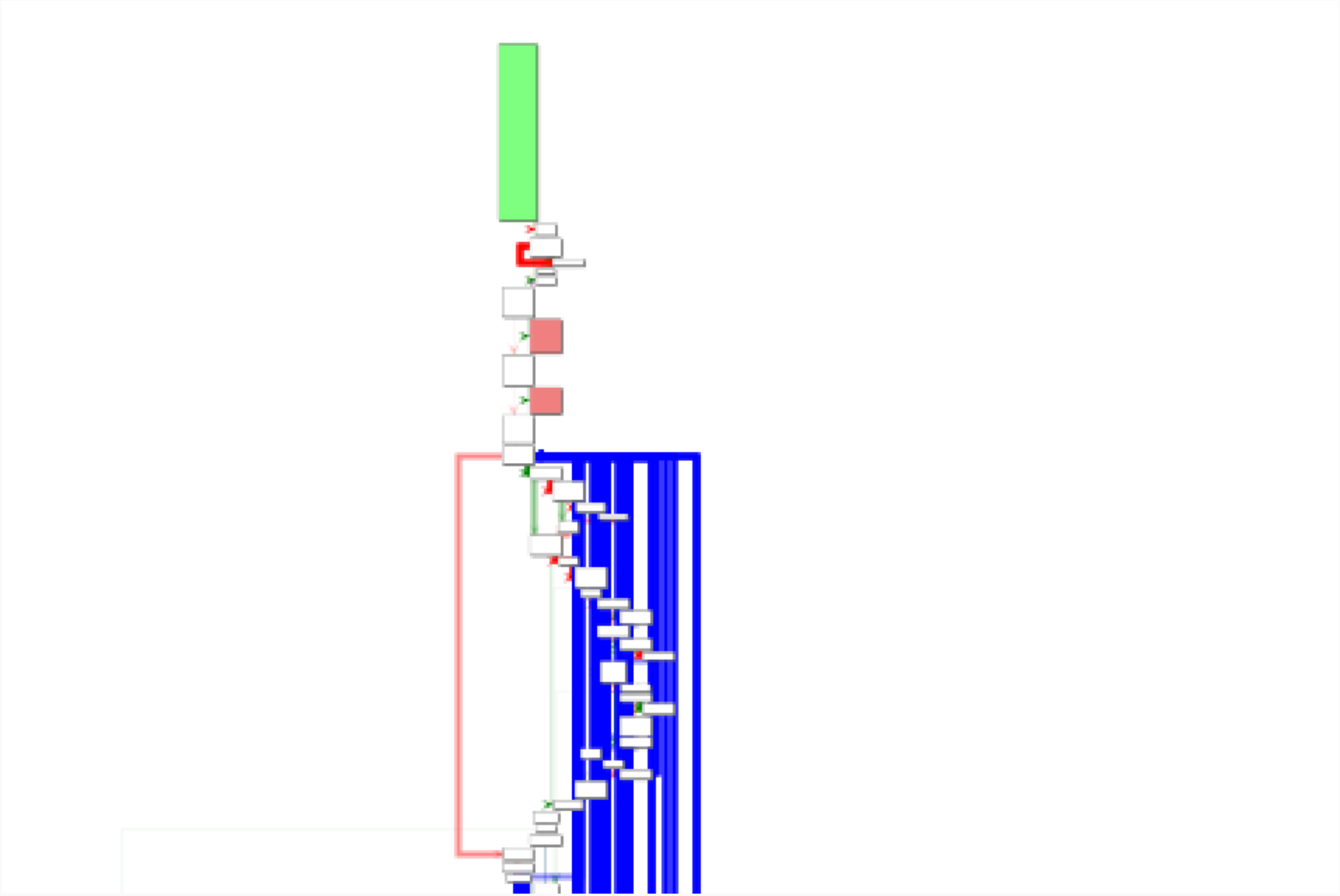


Figure 2: Code Blocks am Beispiel des ssh Binary

Monitoring

- Überwacht das zu untersuchende System
  - Codeabdeckung
  - Laufzeit
  - Speicherverbrauch
  - Crashes
  - ...
- Oft auf Basis von Instrumentierung
- Kann größtenteils über die Laufzeitumgebung realisiert werden

Strategie

- Steuert den Ablauf des Fuzzers
- Bestimmt, welcher Testcase wann verwendet wird
  - Codeabdeckung (AFL) = Wie viele (neue) Basic Blocks werden durch den Testcase traversiert?
  - Zielgerichtet (Dowser) = Spezifische Codepfade, die durch den Testcase erreicht werden sollen

Techniken des Fuzzings

Techniken des Fuzzings

- Mutation (AFL)
- Generation (boofuzz)
- Machine Learning & KI (Pulsar)

Machine Learning/KI und Fuzzing

Aktueller Trend in Fuzzing

- 658 bisher veröffentlichte Paper zum Thema Fuzzing seit 2010 [2]
- 2017 erstes Paper zum Thema Machine Learning [3]
- 2024 bereits 9 Paper zum Thema Machine Learning/KI und Fuzzing unter Verwendung von Reinforcement Learning und LLMs [2]

Machine Learning und Fuzzing

Wieso Machine Learing und Fuzzinng?

- Machine Learning kann interessante Eingaben vorhersagen
- Komplexe Strukturen von Eingaben können erkannt werden
- Verbessertes Verständnis von Code(-fehl-)verhalten
- Zusammenhänge zwischen verschiedenen Programmen und Systemen knüpfen

PULSAR: Stateful Black-Box Fuzzing of Proprietary Network Protocols

Hugo Gascon, Christian Wressnegger, Fabian Yamaguchi,
Daniel Arp, and Konrad Rieck

Computer Security Group
University of Göttingen
{hgascon,christian.wressnegger,fabian.yamaguchi,
darp,konrad.rieck}@uni-goettingen.de

Abstract. The security of network services and their protocols critically depends on minimizing their attack surface. A single flaw in an implementation can suffice to compromise a service and expose sensitive data to an attacker. The discovery of vulnerabilities in protocol implementations, however, is a challenging task; While

Figure 3: Veröffentlichung des ersten Fuzzers mit Markov Modellen [4]

Pulsar

- Erster Fuzzer mit “Machine Learning”
- Verwendung von Markov-Modellen
- Erkennt und simuliert Zustände
- Erkennt und simuliert Nachrichten
- Kann sowohl ein Protokoll simulieren, als auch das tatsächliche Protokoll fuzzen

Anwendungsfälle für Machine Learning und Fuzzing

- Generierung von besseren Eingaben mittels Reinforcement Learning
- Generierung von Eingaben mit Generative Adversarial Networks
- Erlernen von Verhaltensweisen von Programmen mittels Unsupervised Learning
- Adaptives Fuzzing mittels Anormalitätserkennung der Ausgaben des zu testenden Programms
- Umgehen von Sicherheitsmechanismen durch Erlernen von Vermeidungsstrategien gegen Intrusion Detection Systeme

Limitationen und Herausforderungen von ML und Fuzzing

- Hohe Rechenanforderungen
- Trainingsdatenknappheit in neuen und Nischen Domänen
- Model Generalisierung zur Anwendbarkeit auf neue Domänen kann zu ungewolltem Overfitting führen

Ausblick

Fuzzing von Netzwerkprotokollen mit Machine Learning

- Fuzzer kann durch frei verfügbaren Netzwerkverkehr trainiert werden
- Eingaben sollen von bereits existierenden Netzwerkprotokollen abgeleitet werden und darin bereits enthaltene Fehler reproduzieren
- Dynamische Analyse zur Laufzeit des Protokolls soll die Generierung der Eingaben durch Reinforcement Learning verbessern
- Crashes und Bugs sollen gelabelt werden, um die Qualität der Eingaben zu verbessern und die Schwere der Bugs zu klassifizieren

Fragen

Quellen

[1] Sanoop Mallissery and Yu-Sung Wu. "Demystify the Fuzzing Methods: A Comprehensive Survey". In: *ACM Comput. Surv.* 56.3 (Oct. 2023). ISSN: 0360-0300. DOI: 10.1145/3623375. URL: https://doi.org/10.1145/3623375.

[2] Cheng Wen. *Recent Papers Related To Fuzzing*. URL: https://github.com/wcventure/FuzzingPaper.

[3] Patrice Godefroid, Hila Peleg, and Rishabh Singh. *Learn&Fuzz: Machine Learning for Input Fuzzing*. 2017. arXiv: 1701.07232 [cs.AI]. URL: https://arxiv.org/abs/1701.07232.

[4] Hugo Gascon et al. "Pulsar: Stateful Black-Box Fuzzing of Proprietary Network Protocols". en. In: *Security and Privacy in Communication Networks*. Ed. by Bhavani Thuraisingham, XiaoFeng Wang, and Vinod Yegneswaran. Vol. 164. Series Title: Lecture Notes of the Institute for Computer Sciences,

Social Informatics and Telecommunications Engineering. Cham: Springer International Publishing, 2015, pp. 330–347. ISBN: 978-3-319-28864-2 978-3-319-28865-9. DOI: 10.1007/978-3-319-28865-9_18. URL: http://link.springer.com/10.1007/978-3-319-28865-9_18 (visited on 06/26/2024).