UWE Bristol | University of the West of England

Improving Search Space Analysis of Fuzzing Mutators Using Cryptographic Structures

Sadegh Bamohabbat Chafjiri (sadegh2.bamohabbatchafjiri@live.uwe.ac.uk)

Prof. Phil Legg, Dr.Michail-Antisthenis Tsompanas, Prof. Jun Hong

University of the West of England

UWEcyber Gold Award

in association with National Cyber Security Centre

Department for Science, Innovation & Technology

Academic Centre of Excellence in **Cyber Security Education**

Agenda

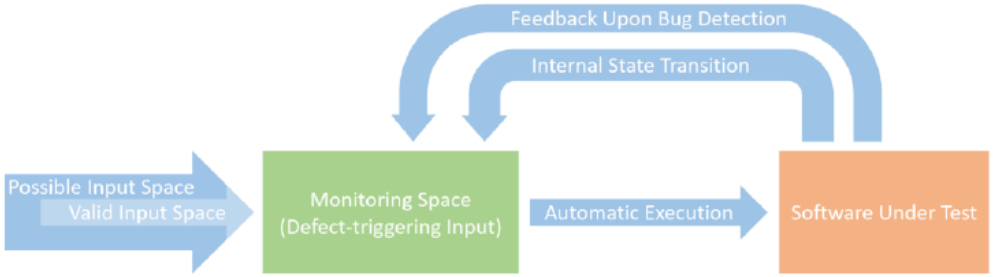
- Introduction
- Mutation-Based Fuzzing
- Challenges in Fuzzing and Research Questions
- Proposed Methodology
- Design of the HonggFuzz+
- Experiment Setup
- Results - Bug Detection Ability and Edge Exploration
- Testing Time Setting
- Future Work and Conclusion
- Acknowledgments and References

Introduction

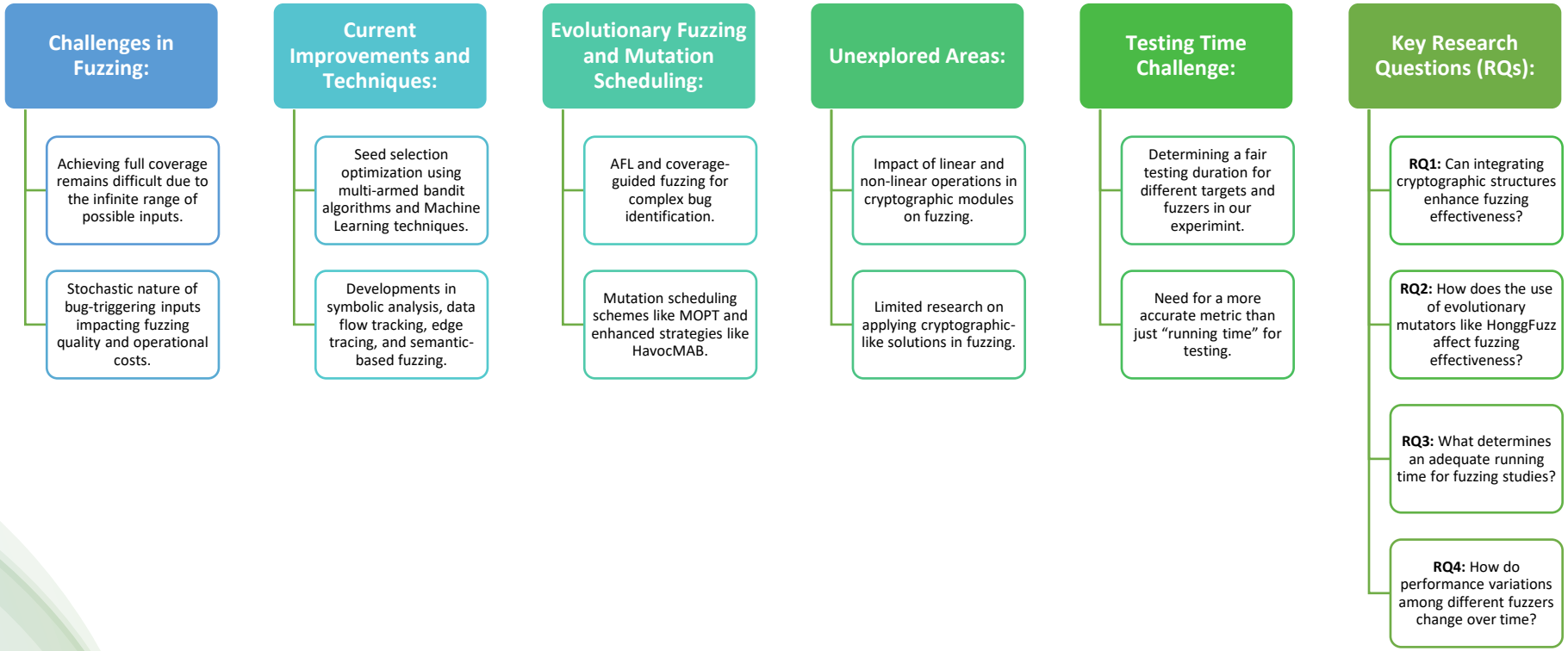
1. **Definition and Purpose:**
    1. Fuzzing: An automated software testing technique.
    2. Focus: Examining software robustness by handling random, unexpected, or malformed inputs.
    3. Goal: Uncover and fix unexpected 'corner cases', potentially leading to security vulnerabilities.
2. **Research Focus:**
    1. Exploring the mutation layer of fuzzing techniques.
    2. Application of the Confusion-Diffusion principle in mutation layers to uncover complex bugs.
3. **This Paper's Contribution:**
    1. Overview of current fuzzing techniques and testing duration measures.
    2. Novel mutation layer adaptation approach using **Substitution-Permutation Networks (SPN) and Feistel Networks (FN)** inspired from the cryptoanalysis domain.
    3. Proposal of a logarithmic curve fitting method for determining testing time.
    4. Comparative performance analysis of new methods against state-of-the-art techniques

Mutation-Based Fuzzing

1. **Mutation-Based Fuzzing:**
    1. Gained popularity for its multi-step process involving bit, byte, and block-level operations (e.g., “seed schedule”, “byte schedule”).
    2. Focuses on reducing the input space to valid inputs while monitoring feedback from outputs.
2. **Evolution to Evolutionary Fuzzing:**
    1. Incorporation of Machine Learning techniques and evolutionary algorithms, like Genetic Algorithm (GA), into fuzzing that offers AFL family.
    2. These advancements led to significant improvements in path exploration efficiency and effectiveness with built-in mutators.
3. **Recent Developments:**
    1. Current research aims to improve the mutation layer to identify complex bugs in fuzzing tools like AFL++ and HonggFuzz.



Challenges in Fuzzing/Research Questions



Proposed Methodology

Integration Objective:

Enhance the ability of HonggFuzz mutator's memory swap function (mangle MemSwap) using SPN and FN.

Aim: Uncover more unique bugs and discover new edges.

Design Philosophy:

Utilize S-box as a non-linear cryptographic primitive for increased input randomness and better edge exploration.

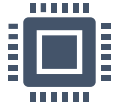
State-of-the-art mutators often use linear memory operations which can be improved with non-linear approaches.

Benefits of Non-Linear Operations:

Introduces dynamic memory rearrangement through confusion property of S-box, enhancing randomization in the fuzzing process.

Increases complexity of test cases, challenging software more effectively.

Aligns with operations in the mathematical domain of Galois Field GF(28), a novel approach in fuzzing which is based on S-box implemented on irreducible multi multinomial



Strengths of the Design:

Non-linearity introduced by S-boxes enhances input confusion and output diffusion.

Improves reliability and efficiency of the fuzzing process.

Diversifies fuzzing strategies, generating high-quality test cases.

Research Focus:

Effectiveness in vulnerability detection through rigorous experimentation

Design of HonggFuzz+

1. **SPHongg:** Substitution layer with S-box and linear operations presented in Figure (a)
2. **FLHongg:** Feistel layer also employing S-box and linear operations presented in Figure (b)

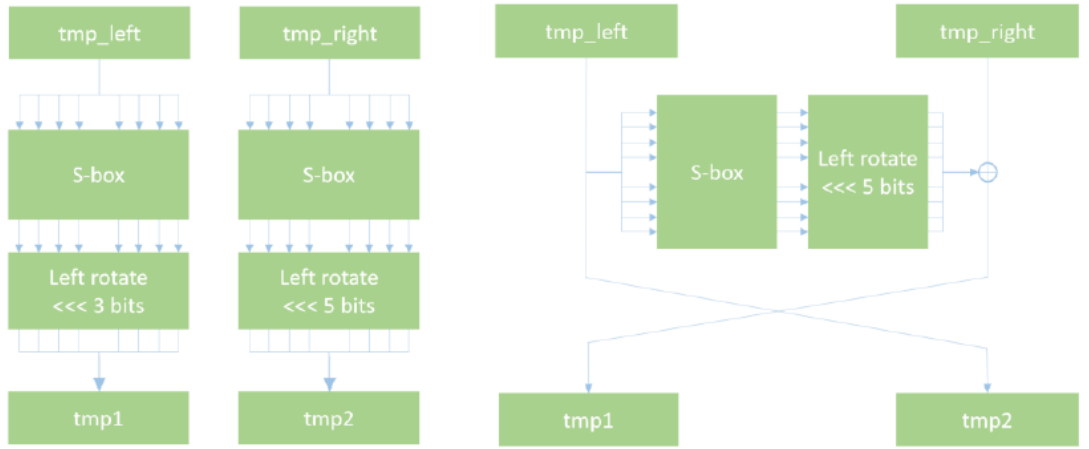
Both structures are integrated into the mangle MemSwap function of HonggFuzz. S-box based on AES (Rijndael) inverse S-box, offering non-linear properties.

Code Block 1 For Loop of SPHongg

```
for (size_t i = 0; i < (len / 2); i++) {
    uint8_t tmp_left = run->dynfile->data[off2 + i];
    const tmp1 = (sbox[tmp_left] << 5) | (sbox[tmp_left] >> (3));
    run->dynfile->data[off2 + i]= run->dynfile->data[off1 + i];
    run->dynfile->data[off1 + i] = tmp1;
    uint8_t tmp_right = run->dynfile->data[off2 + (len - 1) - i];
    const tmp2 = (sbox[tmp_right] << 3) | (sbox[tmp_right] >> (5));
    run->dynfile->data[off2 + (len - 1) - i] = run->dynfile->data[off1 +
    ↪  (len - 1) - i];
    run->dynfile->data[off1 + (len - 1) - i] = tmp2;
}
```

Code Block 2 For Loop of FLHongg

```
 for (size_t i = 0; i < (len / 2); i++) {
   uint8_t tmp_left = run->dynfile->data[off2 + i];
   uint8_t tmp_right = run->dynfile->data[off2 + (len - 1) - i];
   const tmp1 = (sbox[tmp_left] << 5) | (sbox[tmp_left] >>
   ↪  (3))^tmp_right;
   run->dynfile->data[off2 + i] = run->dynfile->data[off1 + i];
   run->dynfile->data[off1 + i] = tmp1;
   const tmp2 = tmp_left;
   run->dynfile->data[off2 + (len - 1) - i] =run->dynfile->data[off1 +
   ↪  (len - 1) - i];
   run->dynfile->data[off1 + (len - 1) - i] = tmp2;
}
```



(a) SPHongg (b) FLHongg

Experiment Setup

Fuzzing Targets:

Focus on document format and image libraries.

Testing conducted on Xpdf, libTIFF, and libexif.

Experiment Setting:

Comparison of Honggfuzz+ mutators with baseline Honggfuzz, AFL++, and LibFuzzer.

Duration: 120 hours (5 days) on three targets.

Platform and Specifications:

Conducted on 10 Kali Linux VM instances, 2GB RAM each.

Host machine: VMware Workstation 16 on Windows 11 Pro, with 10 CPU cores and 64GB RAM.

Mutators Setting:

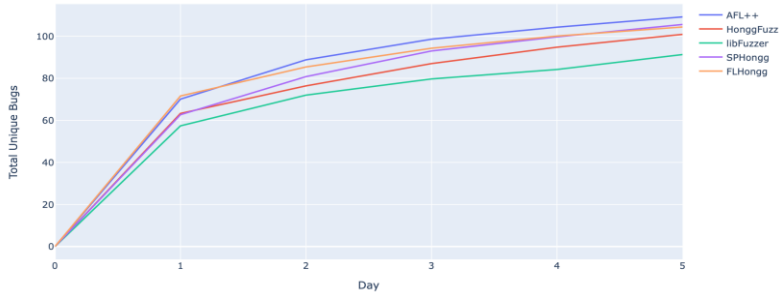
Custom mutator setup in AFL++ using .so files.

Implementation of SPHongg and FLHongg as custom mutators of AFL++.

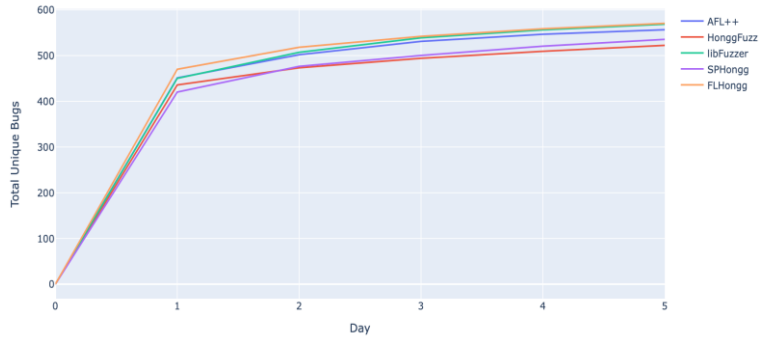
Fixed seed used for all experiments.

Results - Bug Detection Ability and Edge Exploration

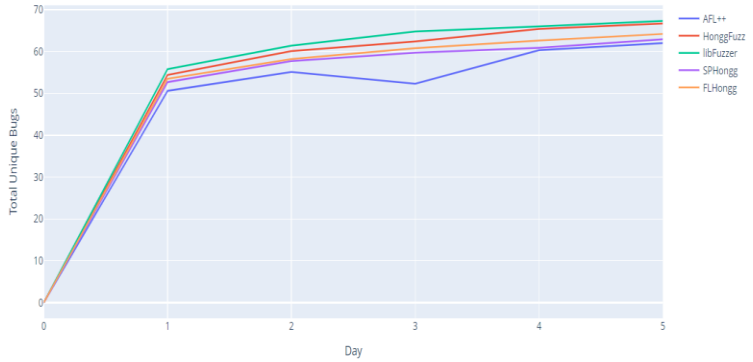
xpdf



libTIFF



libexif



Cumulative Unique Bug Counts Across Five Fuzzers and Three Targets over five days

120-Hour Test Result Overview

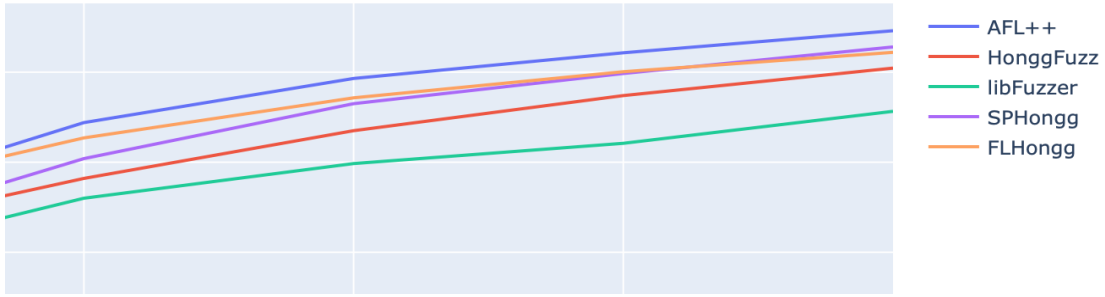
- Cumulative sum of unique bugs
- Edge exploration amount
- Targets: Xpdf, libTIFF, exif

Key Findings of a 120-hour test:

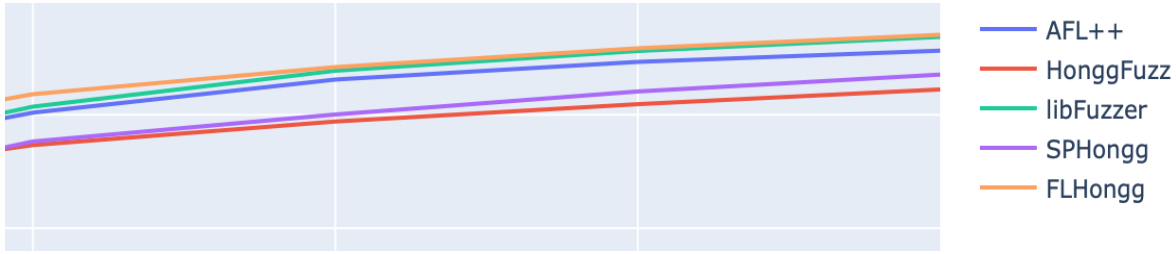
- **HonggFuzz+ structures shows significant improvements in bug detection compared to HonggFuzz on Xpdf and libTIFF targets.**
- Both HonggFuzz+ structures secured second and third ranks on Xpdf, with SPHongg surpassing FLHongg on the last day for the second rank, while FLHongg achieved the first rank on libTIFF and outperformed LibFuzzer on Xpdf and libTIFF.
- On libexif, while LibFuzzer and HonggFuzz had higher ranks, **FLHongg consistently detected bugs close to AFL++.**
- Its adaptable structure can benefit other evolution-based fuzzers like AFL++ and LibFuzzer.

Results - Bug Detection Ability and Edge Exploration

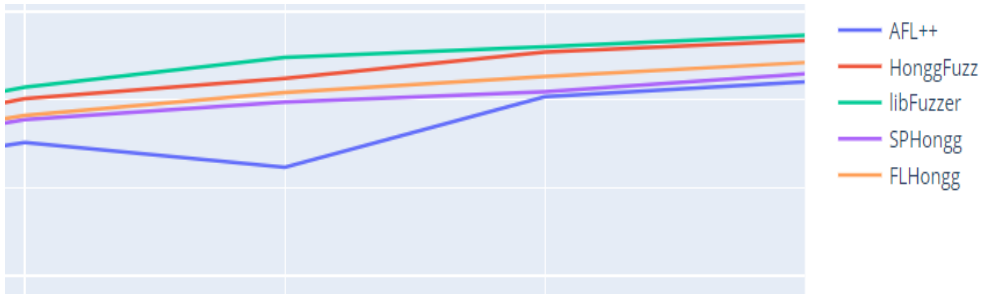
xpdf



libTIFF



libexif



Cumulative Unique Bug Counts Across Five Fuzzers and Three Targets over five days

120-Hour Test Result Overview

- Cumulative sum of unique bugs
- Edge exploration amount
- Targets: Xpdf, libTIFF, exif

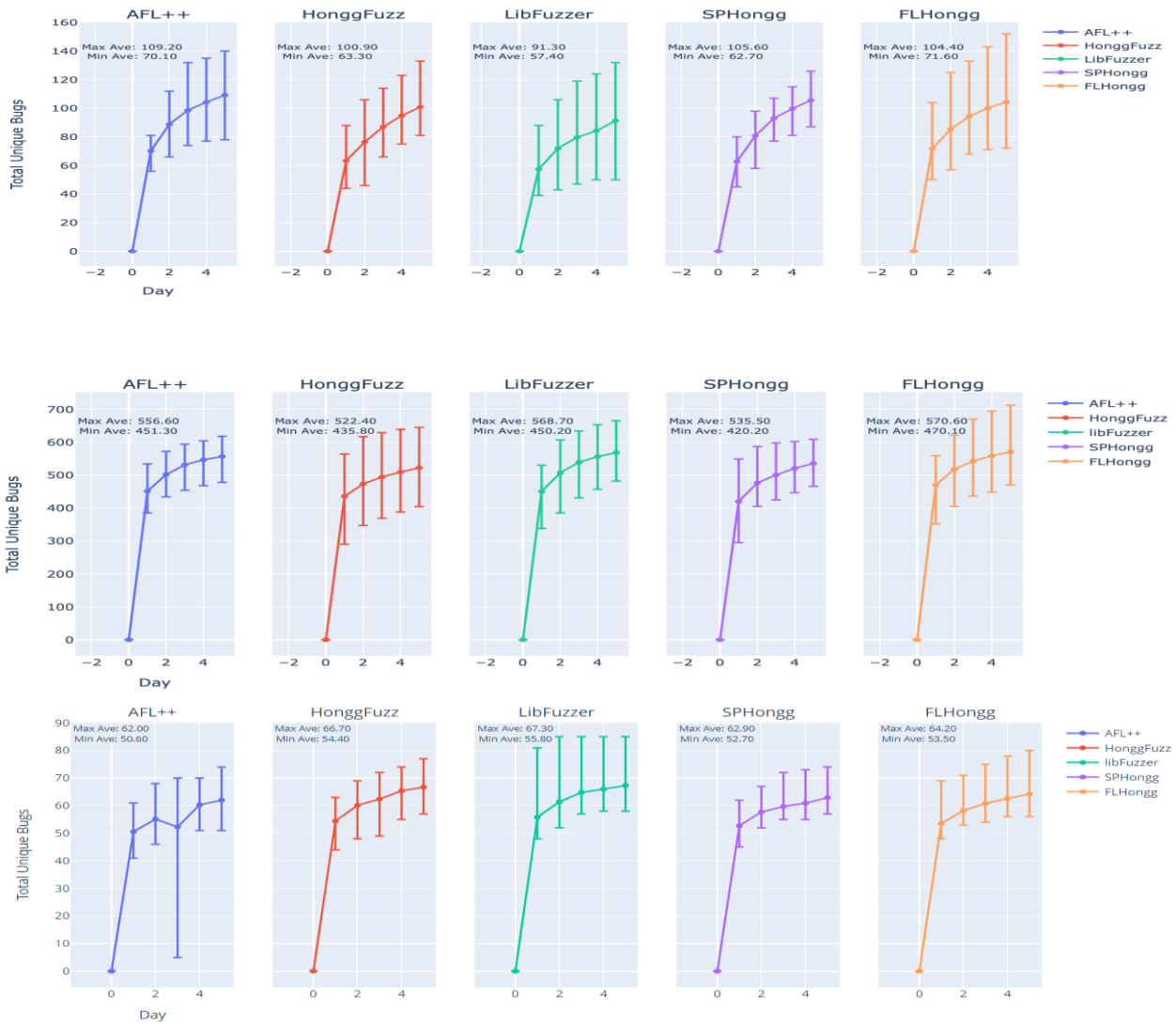
Key Findings of a 120-hour test:

- **HonggFuzz+ structures shows significant improvements in bug detection compared to HonggFuzz on Xpdf and libTIFF targets.**
- Both HonggFuzz+ structures secured second and third ranks on Xpdf, with SPHongg surpassing FLHongg on the last day for the second rank, while FLHongg achieved the first rank on libTIFF and outperformed LibFuzzer on Xpdf and libTIFF.
- On libexif, while LibFuzzer and HonggFuzz had higher ranks, **FLHongg consistently detected bugs close to AFL++.**
- Its adaptable structure can benefit other evolution-based fuzzers like AFL++ and LibFuzzer.

xpdf

libTIFF

libexif



Cumulative Unique Bug Counts Across Five Fuzzers and Three Targets over five days

Results - Bug Detection Ability and Edge Exploration

120-Hour Test Result Overview

- Cumulative sum of unique bugs
- Edge exploration amount
- Targets: Xpdf, libTIFF, exif

Key Findings of a 120-hour test:

- **HonggFuzz+ structures shows significant improvements in bug detection compared to HonggFuzz on Xpdf and libTIFF targets.**
- Both HonggFuzz+ structures secured second and third ranks on Xpdf, with SPHongg surpassing FLHongg on the last day for the second rank, while FLHongg achieved the first rank on libTIFF and outperformed LibFuzzer on Xpdf and libTIFF.
- On libexif, while LibFuzzer and HonggFuzz had higher ranks, **FLHongg consistently detected bugs close to AFL++.**
- Its adaptable structure can benefit other evolution-based fuzzers like AFL++ and LibFuzzer.

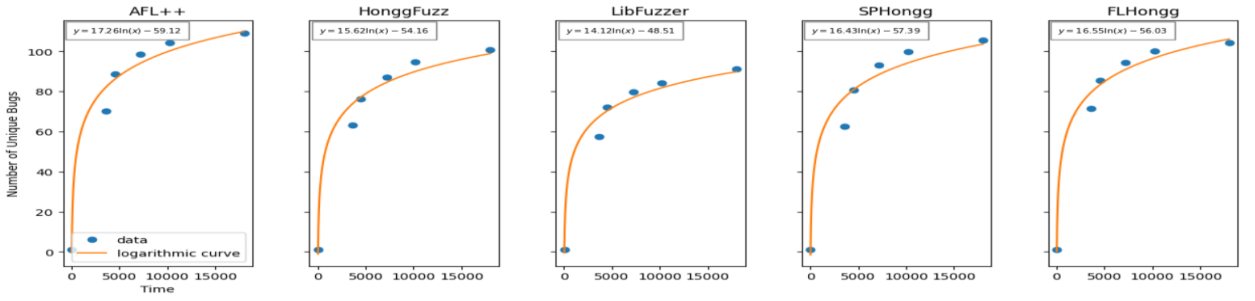
Testing Time

- Our statistical analysis focuses on mutator performance patterns over time.
- We examine cumulative unique bug discoveries during ten fuzzing runs.
- It follows a logarithmic pattern and cumulative sum of number of bugs falls close to or underneath the logarithmic curve on the fifth day on all three targets.

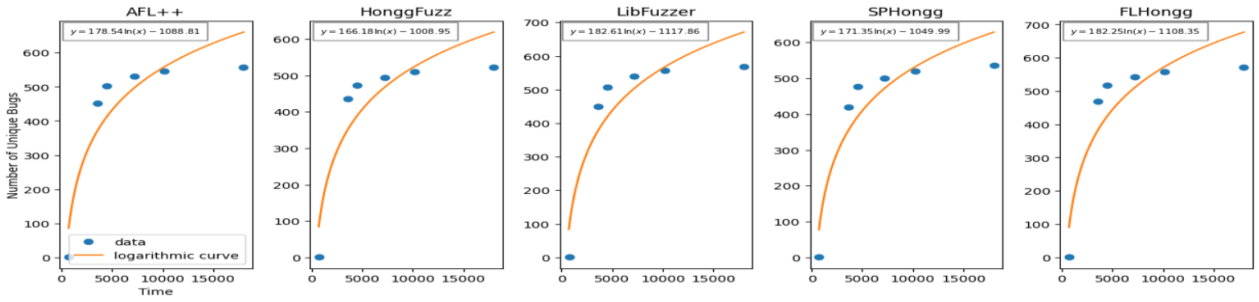
Can a logarithm curve approach help to assess how well the fuzzing performance will continue if a longer test time is used?

Our testing covered the typical 24-hour test as well as up to 120-hour (5 days) testing to assess performance over time.

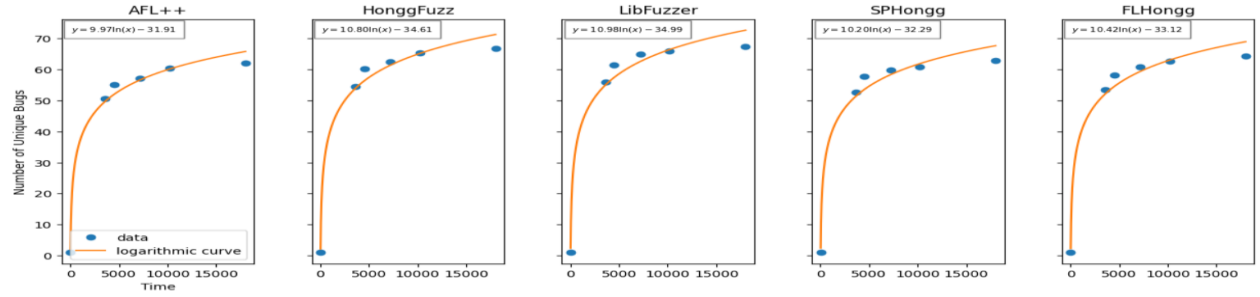
Further testing should be performed for longer durations – as well as suitable termination criteria (e.g., terminate once no new crashes observed within 24 hours).



Xpdf



libTIFF



libexif

Future Work

Broader Target Exploration: Extend the study to encompass a more extensive array of software targets, including standard benchmarks, to evaluate the efficiency and effectiveness of fuzzing approaches across a diverse set of applications.

Benchmark Evaluation: Assess the proposed model's performance against established benchmarks such as Fuzzbench and Magma to provide a comprehensive benchmarking framework.

S-Box Module Expansion: Investigate the potential of expanding the adaptation of the S-box module beyond block-level shuffling to encompass other bit or byte-level operations, enhancing its versatility.

Impact Assessment: Evaluate the impact of incorporating the S-box module in lieu of non-linear operations not only within HonggFuzz but also in the context of other evolutionary fuzzing techniques, contributing to a comprehensive understanding of the benefits of cryptographic integration in fuzzing.

Conclusion

HonggFuzz+, an innovative fuzzer that integrates cryptographic components to optimize the search space, building upon the previous HonggFuzz implementation.

Bug Detection Capabilities: HonggFuzz+ structures, especially FLHongg, demonstrated comparable or superior bug detection capabilities when compared to state-of-the-art evolutionary mutators.

Adaptability: SPN and FN structures showcased adaptability for use in other evolution-based fuzzers, employing a comparable framework of non-linear operations.

Fixed Test Time Issue: We highlighted the limitations of fixed testing durations when comparing different fuzzing approaches, proposing an alternative approach based on evaluating cumulative bug detection over time.

Need for Adaptability: Our findings underscore the necessity for a more nuanced and adaptable evaluation framework for fuzzing methodologies.

Integration Benefits: HonggFuzz+ showcased remarkable bug detection capabilities, emphasizing the potential advantages of integrating cryptographic modules into the software testing and vulnerability detection toolkit.

Thank you for your attention

Sadegh Bamohabbat Chafjiri (sadegh2.bamohabbatchafjiri@live.uwe.ac.uk)

Prof. Phil Legg, Dr.Michail-Antisthenis Tsompanas, Prof. Jun Hong

University of the West of England

UWE cyber Gold Award — in association with National Cyber Security Centre — Department for Science, Innovation & Technology — Academic Centre of Excellence in **Cyber Security Education**