5. Thiết kế và đánh giá thí nghiệm

- Nhận định về ConFuzzius: ConFuzzius là một công cụ fuzz testing kết hợp với symbolic execution. Symbolic execution là kỹ thuật giúp phân tích mã nguồn bằng cách sử dụng các biểu thức tượng trưng (symbolic expressions) thay vì giá trị cụ thể. Công cụ này thu thập và giải quyết các ràng buộc tượng trưng để khám phá các nhánh mã chưa được kiểm tra. Khi các ràng buộc được giải, ConFuzzius tạo ra các test case mới dựa trên kết quả đó và sử dụng chúng để tạo các mẫu thử nghiệm khác, từ đó cải thiện phạm vi bao phủ mã (code coverage)

=> CrossFuzz được xây dựng dựa trên ConFuzzius, bổ sung các chức năng để triển khai nhiều hợp đồng thông minh (multi-contract deployment), cấu hình hàm khởi tạo (constructor settings), và chiến lược đột biến chuỗi giao dịch (transaction sequence mutation)

- Để đánh giá khả năng bao phủ mã và phát hiện lỗ hổng của CrossFuzz thì ta sẽ tập trung vào 3 Research Questions ( câu hỏi nghiên cứu ):

+ RQ1: Liệu chiến lược đột biến chuỗi giao dịch có cải thiện khả năng bao phủ mã không? Nếu có, xác suất tối ưu để kích hoạt chiến lược này là bao nhiêu?

+ RQ2: Liệu CrossFuzz có thể đạt được khả năng bao phủ mã cao hơn và phát hiện lỗ hổng tốt hơn so với các phương pháp fuzz testing hiện đại không?

+ RQ3: Các tham số được tạo cho hàm khởi tạo có cải thiện khả năng bao phủ mã và phát hiện lỗ hổng không?

5.1 Thiết kế thí nghiệm

- Tập trung vào 4 khía cạnh: tập dữ liệu, cài đặt tham số, số liệu đánh giá và môi trường thử nghiệm

\* Tập dữ liệu

- Thực hiện tiến hành thử nghiệm trên **396 hợp đồng thông minh** ( không sử dụng các hợp đồng thông minh mà không cần gọi hàm bên ngoài vì nó không phù hợp để kiểm tra các lỗ hổng liên quan đến tương tác giữa các hợp đồng ), các hợp đồng được chia thành 3 kích cỡ là số lượng của các dòng mã <100, <500 và >= 500

- Dữ liệu thống kê như sau:  
A screenshot of a computer

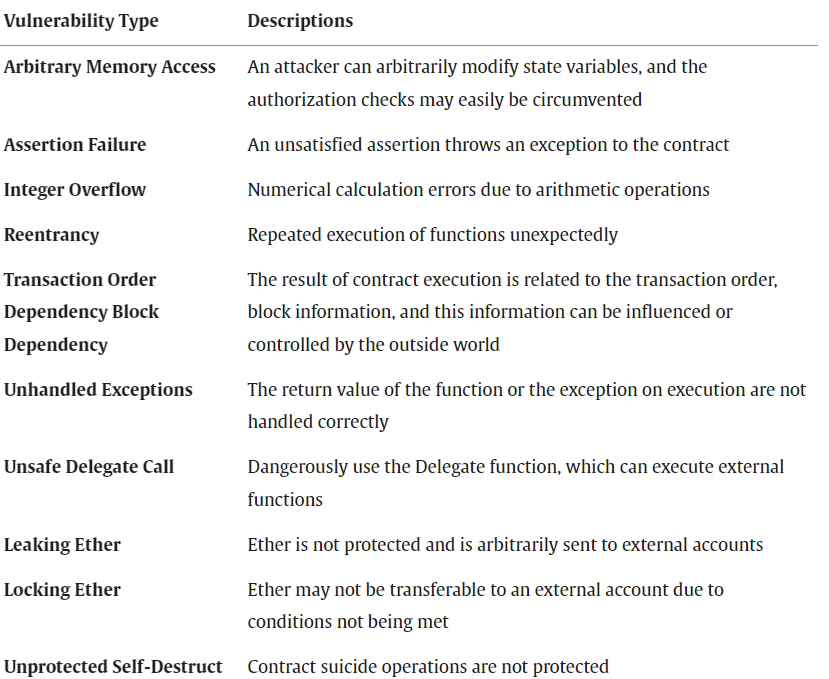
Description automatically generated

\* Chỉ số đánh giá

- Độ bao phủ code: coverage = số lệnh thực thi / tổng số lệnh

- Số lượng lỗ hổng phát hiện được: Tiêu chí này phụ thuộc vào việc dữ liệu kiểm thử có cung cấp nhãn lỗ hổng (vulnerability labels) hay không, tức là dữ liệu cần được gắn nhãn để chỉ rõ hợp đồng nào có lỗ hổng và loại lỗ hổng

- CrossFuzz có thể phát hiện 11 loại lỗ hổng bảo mật như sau:



- Ngoài ra CrossFuzz cũng tính toán độ lệch chuẩn và minh họa nó trên biểu đồ thanh để minh họa ý nghĩa thống kê

\* Cơ sở so sánh

- So sánh CrossFuzz với 3 mã nguồn mở khác về Fuzz là: sFuzz, ConFuzzius, và xFuzz

\* Thiết lập tham số

- Tất cả các nguồn đều sử dụng tham số mặc định

- Đặt thời gian thử nghiệm 1 hợp đồng tối đa là 10 phút

- Để tránh ảnh hưởng của tính ngẫu nhiên đến kết quả thử nghiệm, mỗi hợp đồng thông minh được thử nghiệm năm lần trong thử nghiệm và giá trị trung bình của năm kết quả được lấy làm kết quả thử nghiệm cuối cùng

- Thử nghiệm được thực hiện trên OS Ubuntu 20.04 LTS với Intel Core i7-12700 và bộ nhớ 32 GB

5.2. Đánh giá

I/ RQ1: Liệu chiến lược đột biến chuỗi giao dịch có cải thiện khả năng bao phủ mã không? Nếu có, xác suất tối ưu để kích hoạt chiến lược này là bao nhiêu?

A graph of a number of graphs

Description automatically generated with medium confidence

=> Việc kích hoạt chiến lược đột biến trình tự giao dịch dựa trên ICDF có thể cải thiện khả năng bao phủ mã của CrossFuzz. Sự cải thiện tốt nhất đạt được khi xác suất kích hoạt chiến lược đột biến chuỗi giao dịch là 80%.

II/ RQ2: Liệu CrossFuzz có thể đạt được khả năng bao phủ mã cao hơn và phát hiện lỗ hổng tốt hơn so với các phương pháp fuzz testing hiện đại không?

A graph of different colored bars

Description automatically generated

=> Phạm vi bao phủ của sFuzz, xFuzz và ConFuzzius giảm dần khi quy mô hợp đồng tăng lên, vì mã hợp đồng lớn hơn có xu hướng chứa nhiều kết hợp chuỗi giao dịch hơn, điều này làm giảm hiệu quả của việc làm mờ. Đồng thời, hình cho thấy phạm vi bao phủ của CrossFuzz vượt trội hơn các phương pháp làm mờ khác ở các quy mô hợp đồng khác nhau

A graph of different sizes and colors

Description automatically generated

=> Số lượng lỗ hổng được CrossFuzz phát hiện lớn hơn số lượng lỗ hổng được ConFuzzius phát hiện ở các quy mô hợp đồng khác nhau. Nhìn chung, sau khi tiến hành fuzzing 5 lần trên 396 hợp đồng thông minh, số lỗ hổng bảo mật được CrossFuzz và ConFuzzius phát hiện là 12092 và 6662

A screenshot of a computer

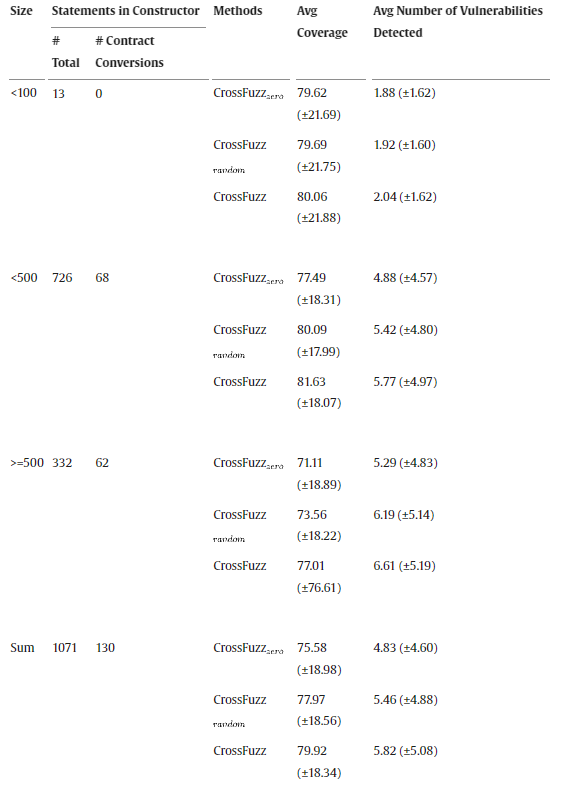
Description automatically generated

=> CrossFuzz phát hiện nhiều lỗ hổng hơn ConFuzzius ở tất cả các danh mục ngoại trừ “Rò rỉ Ether”. Trong số các số liệu này, CrossFuzz có sự cải thiện lớn hơn về các lỗ hổng “tái xuất” và “các ngoại lệ chưa được xử lý”

- Một trong những lý do khiến CrossFuzz phát hiện ít lỗ hổng Leaking Ether hơn ConFuzzius là vì việc kích hoạt các lỗ hổng đó đòi hỏi người gửi hoặc người nhận giao dịch phải đáp ứng các điều kiện cụ thể (ví dụ: người nhận phải là kẻ tấn công được xác định trước). Vì CrossFuzz triển khai nhiều hợp đồng hơn nên việc tạo ra các chuỗi giao dịch cụ thể khó khăn hơn so với ConFuzzius

===> CrossFuzz vượt trội hơn các công cụ kiểm tra fuzz hiện có về phạm vi bao phủ mã hợp đồng và khả năng phát hiện lỗ hổng. Độ bao phủ bytecode trung bình của CrossFuzz được cải thiện 10,58% so với xFuzz và số lỗ hổng bảo mật trung bình được phát hiện lớn hơn 1,82 lần so với ConFuzzius

III/ RQ3: Các tham số được tạo cho hàm khởi tạo có cải thiện khả năng bao phủ mã và phát hiện lỗ hổng không?



- Thấy rằng CrossFuzz hoạt động tốt trong các hợp đồng thông minh có quy mô lớn hơn. Khi kích thước của hợp đồng thông minh lớn hơn hoặc bằng 500

- Sự cải thiện của CrossFuzz tăng lên khi tỷ lệ câu lệnh chuyển đổi hợp đồng trên tổng số câu lệnh trong hàm tạo

======> So với việc sử dụng địa chỉ 0 hoặc ngẫu nhiên để điền các tham số của hàm tạo, CrossFuzz, tạo ra các tham số của hàm tạo bằng cách theo dõi các đường truyền dữ liệu, có thể cải thiện phạm vi bao phủ và số lượng lỗ hổng được phát hiện. Hơn nữa, khi có nhiều câu lệnh chuyển đổi hợp đồng hơn trong hàm tạo của hợp đồng đang được thử nghiệm, hiệu suất của CrossFuzz sẽ được cải thiện hơn so với hiệu suất của tham số hàm tạo điền theo địa chỉ 0 hoặc địa chỉ ngẫu nhiên

5.3. Thảo luận

5.3.1. Tính hợp lệ

\* Tính hợp lệ nội bộ: Đảm bảo rằng CrossFuzz được triển khai đúng cách để kết quả thử nghiệm phản ánh chính xác hiệu quả của công cụ

- Giải pháp:

+ CrossFuzz được phát triển dựa trên các công cụ mã nguồn mở là Slither, Confuzzius và sFuzz nhằm giảm thiểu lỗi và tăng độ tin cậy

=>Mục tiêu: Đảm bảo rằng CrossFuzz hoạt động đúng chức năng và kết quả không bị ảnh hưởng bởi sai sót trong quá trình triển khai.

\* Tính hợp lệ bên ngoài: Liên quan đến khả năng tổng quát hóa kết quả thí nghiệm sang các trường hợp khác

- Giải pháp:

+ Bị hạn chế từ bộ dữ liệu công khai => tạo bộ dữ liệu mới

+ So sánh với các công cụ khác

=>Mục tiêu: Kết quả thử nghiệm có thể áp dụng rộng rãi trong việc phát hiện lỗ hổng hợp đồng thông minh.

\* Tính hợp lệ cấu trúc: Liên quan đến việc sử dụng các thước đo phù hợp để đánh giá hiệu quả của CrossFuzz

- Giải pháp:

+ Sử dụng các thước đo phổ biến là độ bao phủ và số lượng lỗ hổng phát hiện

+ Đánh giá thời gin càn thiết để thực hiện các thử nghiệm

=>Mục tiêu: Các thước đo được chọn cung cấp cái nhìn toàn diện về hiệu quả và độ chính xác của CrossFuzz

5.3.2. Những cải tiến cần thực hiện

1. Giới han chiều dài chuỗi giao dịch

- Vấn đề: Một số nhánh mã chỉ có thể được bao phủ nếu chuỗi giao dịch có độ dài lớn hơn

+ Hiện tại, CrossFuzz giới hạn độ dài chuỗi giao dịch ở mức 5, theo cấu hình mặc định của ConFuzzius

+ Khả năng cải thiện: Tăng độ dài chuỗi có thể giúp tăng độ bao phủ mã nhưng sẽ làm tăng chi phí thời gian kiểm thử fuzz

2. Giá trị mặc định của biến trạng thái

- Vấn đề: Một số nhánh mã chỉ được bao phủ nếu các biến trạng thái trong constructor có giá trị mặc định thỏa mãn điều kiện nhánh

+ CrossFuzz hiện không xử lý tình huống này và các giá trị tham số được tạo theo ICDF có thể làm giảm độ bao phủ mã

+ Khả năng cải thiện: Cung cấp giao diện cho phép người dùng nhập thủ công các giá trị tham số cụ thể cho constructor

3. Hàm cần được gọi nhiều lần

- Vấn đề: Một số nhánh mã chỉ được bao phủ nếu một số hàm được gọi nhiều lần

+ CrossFuzz hiện chỉ chọn các hàm chưa được gọi trước đó để thêm vào chuỗi giao dịch, nên có thể bỏ lỡ các nhánh này

+ Khả năng cải thiện: Cho phép người dùng thay đổi chiến lược này thông qua cài đặt tham số

4. Gọi hợp đồng bên ngoài qua địa chỉ

- Vấn đề: CrossFuzz chưa hỗ trợ kiểm thử các tình huống khi hợp đồng gọi các hợp đồng khác thông qua địa chỉ

+ Khả năng cải thiện: Trong tương lai, nhóm phát triển dự kiến sử dụng EtherScan để tìm kiếm mã nguồn của các hợp đồng bên ngoài, mở rộng khả năng áp dụng của CrossFuzz