

Security Code Recommendations for Smart Contract

Xiaocong Zhou¹, Yingye Chen¹, Hanyang Guo², Xiangping Chen³, Yuan Huang^{2,*} ¹School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China ²School of Software Engineering, Sun Yat-sen University, Zhuhai, China
³School of Communication and Design, Sun Yat-sen University, Guangzhou, China

発表者：野口 優輝

スマートコントラクト

- スマートコントラクト

- ブロックチェーン上で動作し、契約を自動的に実行する仕組み
- 高水準のプログラミング言語Solidityを用いて開発される
- 資産の取引などはスマートコントラクトに依存している



- 近年では、スマートコントラクトのセキュリティでの脆弱性を悪用し、セキュリティ事故^{[1][2]}などが頻繁に起こっている。

[1] Xiangfu Zhao, Zhongyu Chen, Xin Chen, Yanxia Wang, and Changbing Tang. The dao attack paradoxes in propositional logic. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 1743–1746. IEEE, 2017.

[2] Giuseppe Destefanis, Michele Marchesi, Marco Ortu, Roberto Tonelli, Andrea Bracciali, and Robert Hierons. Smart contracts vulnerabilities: a call for blockchain software engineering? In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 19–25. IEEE, 2018.

既存研究

- 動的, 静的解析に基づくスマートコントラクトの脆弱性回避ツールがある
 - ファジング技術^{[3][4]}, 記録実行^[5]. . .
- これらの技術の問題点
 - セキュリティコードの推奨事項を提供できない
 - スマートコントラクトのプログラム自動修復できない

[3] Jingxuan He, Mislav Balunovic^{*}, Nodar Ambroladze, Petar Tsankov, and Martin Vechev. Learning to fuzz from symbolic execution with application to smart contracts. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 531–548, 2019.

[4] Chao Liu, Han Liu, Zhao Cao, Zhong Chen, Bangdao Chen, and Bill Roscoe. Reguard: finding reentrancy bugs in smart contracts. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pages 65–68. IEEE, 2018.

[5] Johannes Krupp and Christian Rossow. {teEther}: Gnawing at ethereum to automatically exploit smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1317–1333, 2018.

提案手法

- 統計学習に基づくスマートコントラクトの自動プログラム修復を探索
- 2000のスマートコントラクトのOSプロジェクト
 - プログラム修復に関する24153のメソッドペアを抽出
 - コード変更の範囲が1行であるものは12219件と半数以上
- 一行パッチを用いたプログラム修復に関するデータセットに着目

提案手法

- 多くのスマートコントラクトの脆弱性
 - バギーラインにおける，条件判定やアクセス制御に関する記述と関連
 - Reentrancy脆弱性
 - スマートコントラクト史上最も深刻な脆弱性
 - バギー行の次の行で，問題が発生する
 - バグが発生した次の行を前の行に移すと解決
 - 1行パッチを効率よく生成するには，バグの前後を含めた3行のコンテキストを十分に考慮する必要
- 
- SMARTREPを提案
 - 1行パッチを用いたスマートコントラクト修復のアプローチ

バギー：
バグを含んでいる

アプローチ

- データ収集

- Githubからスマートコントラクトに関するプロジェクトを収集

- スター上位2000件を収集（Solidity：1000件， javascript：1000件）

- バグ関連のコミットリストを収集

- コミットメッセージに「fix」，「bug」，「resolve」，「patch」が含まれるもののみを収集

- 統計学習

- 1行パッチを生成するために，ダブルエンコーダーネットワークを採用

- 1つ目のエンコーダー：手法全体の情報を取得

- 2つ目のエンコーダー：アクセス制御，条件判定文に関する情報を取得



スマートコントラクト脆弱性に関連

RQ

- RQ1 :

- 正しい1行パッチに関して, SMARTREPの性能は,
従来の統計的学習ベースの修復技術と比較してどうか

- RQ2 :

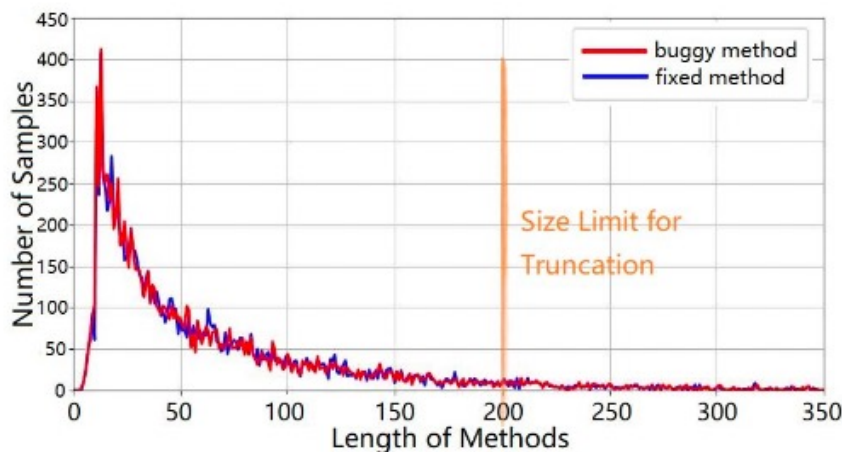
- 第2のエンコーダーは, プログラム自動修復の効率を
どの程度向上するか

- RQ3 :

- どのような種類のスマートコントラクトにおける脆弱性を
修正できるか

データセット

- 長さの異なるバギーコードを修正するSMARTREPを評価
- Small, Medium, Totalのデータセットを用いる
 - Small : 50未満のメソッドを含む
 - Medium : 50以上, 200未満のメソッドを含む
 - Total : 200未満のメソッドを含む



200以上のメソッドはほとんどないことがわかる

評価 | RQ1：正しい1行パッチ

- データセット
 - 8：2の割合で訓練セットとテストセット
- 既存の修復技術
 - SequenceRを使用（1行パッチのプログラム手法の中では最新鋭）
- スマートコントラクトの修復の有効性を測定
 - structurally：正しいコード構造になっているか
 - completely：コード修復がどのくらい完全か

RQ1 結果

	Small		Medium		Total	
	<i>structurally</i>	<i>completely</i>	<i>structurally</i>	<i>completely</i>	<i>structurally</i>	<i>completely</i>
<i>seq2seq(line2line)</i>	135/1,234 (10.1%)	97/1,234 (8.0%)	45/992 (4.5%)	24/992 (2.4%)	215/2,248 (9.6%)	150/2,248 (6.7%)
<i>Tufano et al. [23]</i>	281/1,234 (22.8%)	162/1,234 (13.1%)	187/992 (18.9%)	86/992 (8.7%)	407/2,248 (18.1%)	213/2,248 (9.5%)
<i>SequenceR [24]</i>	390/1,234 (31.6%)	264/1,234 (21.4%)	316/992 (31.8%)	141/992 (14.2%)	769/2,248 (34.2%)	487/2,248 (21.6%)
SMARTREP	464/1,234 (37.6%)	304/1,234 (24.6%)	275/992 (27.7%)	147/992 (14.8%)	966/2,248 (42.9%)	535/2,248 (23.8%)

従来 of 技術との比較

評価 | RQ2: 第2のエンコーダー

- 第2エンコーダーが捉える有用性を理解
 - アブレーション実験を行う

Input to the Second Encoder	Total
Nothing (remove the second encoder)	300/2,248 (13.3%)
Previous line & Buggy line	488/2,248 (21.7%)
Buggy line & Next line	507/2,248 (22.5%)
Two previous line & Buggy line & Two next line	387/2,248 (17.2%)
Previous line & Buggy line & Next line (SMARTREP)	535/2,248 (23.8%)

アブレーション実験
構成要素の一部を取り除いて、結果を比較すること

- 第2のエンコーダーがSMARTREPと比べて、10.5%ほど減少している
 - 第2のエンコーダーはプログラム修復に重要な役割を持っている

評価 | RQ3 : 脆弱性修復

- 典型的なスマートコントラクトの脆弱性を分析
 - SCRepairとSMARTREPを比較

	[10]	SCRepair [9]	SMARTREP
Use of Deprecated Functions	-	-	65/101
Unchecked Return Value	✓	7/17	8/17
Incorrect Inheritance Order	-	-	12/55
Illegal Coverage	-	-	5/26
Transaction Order Dependence	-	0/3	0/3
Reentrancy	✓	2/21	6/21
Erroneous Visibility	-	-	79/214
Arithmetic Issue	✓	0/90	23/90
Missing Return Statement	-	-	1/18
Authorization Through tx.origin.	-	-	9/14
Erroneous Variable Type	-	-	48/111
Others	-	-	279/1578
Total	-	9/131	535/2248

131件の脆弱性を実行するのに

SCRepair→27日
SMARTREP→3時間



SMARTREPはより迅速に
修復することができる

結論

- スマートコントラクトにはセキュリティ上の脆弱性がある
- 脆弱性を解決するため、SMARTREPを提案
- SMARTREP
 - 1行パッチを用いたスマートコントラクト修復のアプローチ

結果

- 既存の自動プログラム修復技術よりもSMARTREPは多くの種類の脆弱性を修正することができた