Communications of the CCISA
Vol. 29 No. 1 Feb. 2023

智能合約的安全防護與檢測平台實作

江毓晴[1]、陳元娣[2]、呂明憲[3]、汪侑翰[4]、陳宗和[5]
[1,2,3,4,5]國立嘉義大學資訊工程學系
s1072928@g.ncyu.edu.tw[1]、s1072911@g.ncyu.edu.tw[2]、s1082955@g.ncyu.edu.tw[3]、
s1082949@g.ncyu.edu.tw[4]、thchen@mail.ncyu.edu.tw[5]

摘要

以太坊 (Ethereum) 為支持智能合約 (Smart Contract) 最大的區塊鏈 (Blockchain) 平台，透過發佈智能合約的 Bytecode 到鏈上來佈署智能合約之後，就沒辦法再進行更改，所以開發人員在佈署之前檢測智能合約的安全極其重要。目前有許多智能合約的漏洞檢測工具可供給開發人員使用，以確保智能合約的安全，但是這些工具尚未全面性檢測漏洞，且各種攻擊手法與時俱進的在更新，開發人員更容易遺漏各種攻擊的可能性，為了讓開發人員檢驗自己的智能合約之安全性，更全面性的檢測工具是必要的。本研究實作具全面性檢測的智能合約漏洞檢測工具 (稱為 Ladybugs)，除了將 coverage rate 從 55%提升至 94%外，也提升 precision rate、recall rate 至 80%以上。接著以 Ladybugs 為基礎上實作一個具有漏洞介紹的動態掃描區塊鏈智能合約之弱點檢測平台-ContractPecker，透過動態抓取區塊鏈已發佈合約進行檢測與回報，以幫助開發者 (或使用者) 了解合約出現哪些漏洞，介紹漏洞產生原因，以期避免漏洞的出現。

關鍵詞：以太坊、區塊鏈、智能合約、漏洞檢測

Regular Paper

Communications of the CCISA
Vol. 29 No. 1 Feb. 2023

Smart contract security protection and detection platform implementation

Yu-Qing Jiang[1], Yuan-Di Chen[2], Ming-Hsien Lu[3], Yu-Han Wang[4], Tzung-Her Chen [5]
[1,2,3]Department of Computer Science and Information Engineering, National Chiayi University
s1072928@g.ncyu.edu.tw[1]、s1072911@g.ncyu.edu.tw[2]、s1082955@g.ncyu.edu.tw[3]、
s1082949@g.ncyu.edu.tw[4]、thchen@mail.ncyu.edu.tw[5]

Abstract

Ethereum is the largest Blockchain platform that supports Smart Contracts. After deploying smart contracts by publishing Bytecode of smart contracts to the chain, they are irreparable. It is important to check the security of smart contracts before deploying. There are currently many smart contract vulnerability detection tools available to developers to ensure the security of smart contracts. However, these tools have not comprehensively detected vulnerabilities while various attack methods are being updated with the times. It is easier for developers to miss the possibility of various attacks. This study implements a smart contract vulnerability detection tool (called Ladybugs) with comprehensive detection. In addition to increasing the coverage rate from 55% to 94%, it also increases the precision rate and recall rate to more than 80%. Furthermore, a web platform, ContractPecker, is implemented based on Ladybugs by providing a dynamic weakness-detection mechanism to scan the smart contract deployed on Ethereum blockchain. With vulnerability introduction on the basis of Ladybugs Weakness Detection.

Keywords: Ethereum, blockchain, smart contract, vulnerability detection

壹、前言

區塊鏈 (Blockchain) [5]是一種分散式網路，常見的平台有比特幣 (Bitcoin) [1][9]及以太坊 (Ethereum) [7]，在現今可以說是眾所皆知的，並且越來越受到大家的關注。而數位貨幣是目前最熱門的應用，以比特幣為例，在 2021 年 11 月時達到最高匯率 66,311 美元，超越了近期股價大漲的台積電及三星，僅次於蘋果、特斯拉等知名公司，但是比特幣的暴漲和暴跌也嚴重的影響到全世界很多人的財富。

除了用於加密貨幣的交易之外，在 Dapp (Decentralized Application) [10]的應用也越來越普及，並且有各式各樣的應用，最常應用在博奕遊戲以及去中心化交易所，相對於過往運行在中心化服務的 App，Dapp 的程式佈署在區塊鏈上，所有的資料都公開透明且不可竄改，以遊戲為例，每個人都可以自由檢視遊戲的程式碼與規則，解決玩家對於遊戲信任與防止作弊的問題，讓大家可以公平的進行遊戲，又因為 Dapp 是基於區塊鏈去中心化的特性，所以不需要仰賴任何中央伺服器的運作，可以全自動化進行，不需要擔心伺服器受到攻擊而導致癱瘓。而智能合約 (Smart Contract) 就是連結 Dapp 和區塊鏈的橋梁。

目前支持智能合約的最大區塊鏈平台為以太坊，以太坊的創立起因為比特幣平台不具有圖靈完備的腳本系統，Vitalik Buterin 提出改善建議但未得到比特幣平台的同意，所以決定自己開發一個可以開發程式的區塊鏈平台。透過將智能合約的 Bytecode 發佈到以太坊的鏈上來佈署智能合約，也可以說智能合約其實就是放在區塊鏈上的程式碼，利用發起交易的方式與智能合約互動、執行合約中的函式。

智能合約為區塊鏈技術的延伸，具有公開透明、不可竄改、去中心化等特性，如上述所說，基於去中心化特性，不需依賴中央伺服器，也不必擔心伺服器遭到攻擊造成癱瘓，但是基於不可竄改之特性，讓智能合約一旦佈署到區塊鏈上就無法對漏洞進行修補，又因區塊鏈的公開透明，所有人皆可看到程式碼以及 Bytecode (圖 1)，若是被惡意人士發現，仍可利用其漏洞造成財物損失或是讓智能合約癱瘓，所以開發人員必須在佈署之前完整審視智能合約的安全性。

不幸的是，這些漏洞通常難以避免，至今以來，有很多智能合約因而遭受巨大損失，著名事件有 DAO (Decentralized Autonomous Organization) [6]的 Re-entrancy Attack，當時 DAO 因這個漏洞遭竊取價值 7000 萬美元的以太幣 (Ether)，所以在佈署前完善地檢測漏洞是很重要的。


Contract Source Code (Solidity)

```
pragma solidity ^0.4.24;
contract class32{
    address owner;
    constructor() public payable{
        owner = msg.sender;
    }

    function querybalance() public view returns(uint){
        return owner.balance;
    }

    function contractbalance() public view returns(uint){
        return address(this).balance;
    }

    function send(uint money) public returns(bool){
        bool reuslt = owner.send(money);
        return reuslt;
    }

    function transfer(uint money) public {
        owner.transfer(money);
    }
}
```

Contract ABI

```
[{"constant":false,"inputs":[{"name":"money","type":"uint256"}],"name":
[],"name":"contractbalance","outputs":[{"name":"","type":"uint256"}],"p
[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view"
[{"name":"","type":"bool"}],"payable":false,"stateMutability":"nonpayab
```

圖 1：Etherscan 上公開的智能合約內容

DAO 是一個開源 (Open Source) 的智能合約，任何人都可以用以太幣交換 DAO token，透過這種交換方法，為 DAO 籌集了大量的資金。擁有 DAO tokens 的參與者可以進行投票並獲得獎勵，但是 DAO 的智能合約有一個嚴重的缺陷，攻擊者若是在更新餘額前多次向智能合約請求資金，會使攻擊者可以一直取出資金，直到 gas fee (手續費) 不夠或者合約內沒有餘額。該漏洞是由於開發人員未考量會有遞迴呼叫的存在而發生的，使攻擊者能夠在一開始幾個小時內竊取價值數百萬美元的以太幣，這顯示出了簡單的智能合約漏洞可能具有很大的危險。

以太坊有兩種類型的地址 (Account)，一種是 external owed account (EOA)，一種是智能合約地址，而部署智能合約可由 EOA 或是另一個智能合約來完成，具有可執行的程式碼。而交易也分為兩種，一種是發送者為 EOA，另一種是發送者為智能合約。而當呼叫 DAO 的地址屬於智能合約時，DAO 回傳 ether 給智能合約就會被動觸發 fallback 函式。當 callee (呼叫方) 呼叫 caller (被呼叫方) 時，若 caller 有回傳 ether 就會觸發到 callee 的 fallback 函式，此時就可以在函式內再次呼叫 DAO，再次要求取出 ether，呈現遞迴呼叫，這個漏洞是智能合約最危險的漏洞之一，攻擊者可以將智能合約的餘額全部竊取走。

DAO 安全漏洞事件引發大眾對智能合約檢測的重視。我們的研究發現，目前很多智能合約檢測工具還未能夠全面性的偵測。根據我們測試的多種資料集，JiuZhou[8]相較於其他開源的資料集提供了最多種類的漏洞樣本以及更新的版本 (表 1)，所以我們使用 JiuZhou 來當作評估工具能力的基準。但是 JiuZhou 這篇論文中提及的九種智能合約檢測工具，沒有一項工具的 coverage rate 超過 60%，coverage rate 為所有漏洞種類中有被測出來的比例，其中以 Slither 的 55%為最高 (圖 2)，所以我們打算改善 Slither 提升 coverage rate，打造出更具有全面性檢測的檢測工具，加強避免智能合約的不安全導致後續嚴重的負面影響或衝擊。也惟有如此，區塊鏈技術及 Dapp 才能真正落實到應用面。

JiuZhou 是目前包含最多類型漏洞的資料集，裡面提到 49 種漏洞，而且還分成正例與反例。根據我們的測試研究，Slither 只能偵測出 JiuZhou 中 55%的漏洞，但是經過我們改良可以提升至 92%，我們將改良過的檢測工具取名為 LadyBugs。

為了檢驗 LadyBugs 的效能，我們需要大量的智能合約樣本來保證檢測結果的正確性，雖然 JiuZhou 包含的漏洞類型最多，但是每一種漏洞只有 1~3 個智能合約做檢測，數量不足以正確分析效能，所以我們再採取其他兩個資料集來補足，分別為 SolidiF i[4]、SmartBugs[3]，三個資料集總共有 660 個智能合約且分類為 49 種漏洞類型，經過我們的測試，LadyBugs 的 coverage rate 達 92%， precision rate 和 recall rate 也達到 80%以上。

表 1：比較 JiuZhou 與其他資料集的 Solidity 版本

資料集	總合約數量	漏洞種類	Solidity 版本
JiuZhou	176	49	0.4.26~0.6.2
ethernaut	21	21	0.4.18~0.4.24
not-so-smart-contract	25	12	0.4.9~0.4.23
SWC-registry	114	33	0.4.0~0.5.0
capturetheether	19	6	0.4.21

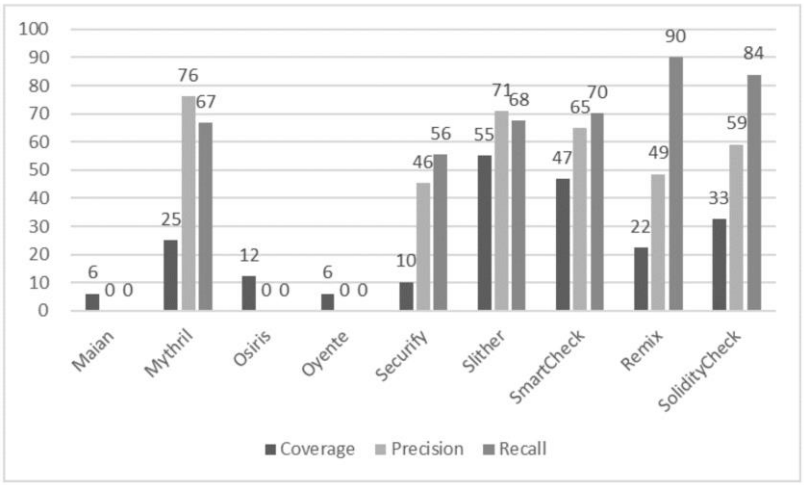

圖 2：各種智能合約檢測工具的 coverage、precision、recall rate (百分比) [8]

多數漏洞的出現是因為開發者對漏洞不夠熟悉，因此我們為平台新增了漏洞介紹功能，包含介紹漏洞成因、影響、正反合約及建議。除此之外，許多智能合約可能還沒經過詳細的檢測就發佈到區塊鏈上，我們會新增動態掃描區塊鏈上智能合約的功能，可以動態抓取區塊鏈已發佈合約進行檢測，幫助開發者了解最新一批智能合約容易出現哪些漏洞，再透過漏洞介紹了解成因及影響，並根據建議避免漏洞的出現。

貳、文獻探討

2.1 資料集

如前所述，我們以 JiuZhou 資料集作為各種檢測工具測試的對象，正因為 JiuZhou 提供 49 種漏洞，擁有最多種漏洞類型，每一種漏洞也都有正、反例可以參考，非常適合拿來學習，但是每一種漏洞只有 1 到 3 個智能合約，總合約樣本數為 176 個，不足用來檢測工具效能，所以我們以以下兩個資料集來補足。

SolidiFi 提供 7 種漏洞的資料集，分別為 Re-entrancy、Timestamp dependency、Unchecked send、Unhandled exceptions、TOD、Integer overflow/underflow、Use of tx.origin，而且在每一種漏洞都有 50 個智能合約樣本，總合約樣本數達到 350 個，是我們蒐集的三個資料集中樣本數最多的。

SmartBugs 提供 10 種漏洞，分別為 Re-entrancy、Access Control、Arithmetic、Unchecked Low Level Calls、Denial Of Service、Bad Randomness、Front Running、Time Manipulation、Short Addresses、Unknown Unknowns，各類型漏洞的樣本較多元，像是以溢位漏洞 (Arithmetic) 來說明，又分別會有向上溢位、向下溢位，向上溢位又分為乘法溢位或是加法溢位，向下溢位分為除法溢位或是減法溢位，而每一種合約樣本數不等，總合約數為 137 個。

2.2 各種檢測平台的比較

如圖 2 所示，在根據 JiuZhou 資料集的測試後，發現 Slither 的表現最平均，其 recall rate 為 68%、precision rate 為 71%，也是文獻中提及的九種檢測工具能測出最多漏洞類型的分析工具。

根據我們的測試，我們將 JiuZhou 分類出的 49 種 bugs 輸入到 Slither 中進行檢測後，發現 Slither 只能偵測出 27 種。我們用表 2 來簡要表示我們的測試結果，如果 Slither 能偵測到的 bug 為 O，反之則為 X。以我們的測試，Slither 的 coverage rate 為 55%。

表 2：檢測平台 Slither 可偵測資料集 JiuZhou 49 種漏洞中的 27 種

漏洞種類	偵測
1. Unintialized Storage Variables	O
2. Unintialized Local/state Variables	O
3. Wrong Operation	O
4. Integer Sign	X
5. Integer Division	O

6. Integer Overflow and Underflow	X
7. Integer Truncation	X
8. Hidden Built-in Symbols	O
9. Hidden State Variables	O
10. Incorrect Inheritance Order	X
11. Right-To-Left-Override Control Character	O
12. Delete Dynamic Array Elements	X
13. Using continue-statements in do-while-statements	X
14. Re-entrancy Vulnerability	O
15. Unhandled Exception	O
16. Forced To Receive Ether	O
17. Locked Ether	O
18. Pre-sent Ether	O
19. Call/delegatecall Data/address Is Controlled Externally	O
20. Hash Collisions with Multiple Variable Length Arguments	X
21. Short Address Attack	X
22. Signature With Wrong Parameter	X
23. Nonstandard Token Interface	O
24. Returning Results Using Assembly Code in The Constructor	O
25. Specify Function Variable as Any Type	O
26. Transaction Order Dependence	X
27. Results Of Contract Execution Affected by Miners (timestamp & random)	O
28. DOS By Non-existent Address or Malicious Contract	X
29. DOS By Complex Fallback Function	X
30. DOS By Gaslimit	O
31. Storage Overlap Attack	X
32. Invariant State Variables Are Not Declared Constant	O
33. Byte[]	X
34. Invariants in Loop	X
35. Unused Public Functions within The Contracts Should Be Declared External	O
36. Wrong Constructor Name	X
37. Use ts.origin For Authentication	O
38. Replay Attack	X
39. Suicide Contract	O
40. Wasteful Contracts	X

41. Non-Public Variables Are Accessed by Public/external Functions	X
42. Public Data	X
43. Improper Use of Require, Assert, and Revert	X
44. View/constant Function Changes Contract State	O
45. Unlimited Compiler Versions	O
46. Implicit Visibility Level	X
47. Nonstandard Naming	O
48. Too Many Digits	O
49. Use Deprecated Built-in Symbols	O

Slither[2]是一個針對 Solidity 的智能合約漏洞檢測工具，執行速度非常快，準確性也非常高，並且可自動檢測漏洞、自動檢測優化、繪製合約繼承圖、函式呼叫圖等等幫助開發員理解程式碼，但是它的缺點是無法準確的分析低階資訊，像是 gas fee 的計算。以下圖 3 為 Slither 程式架構圖：

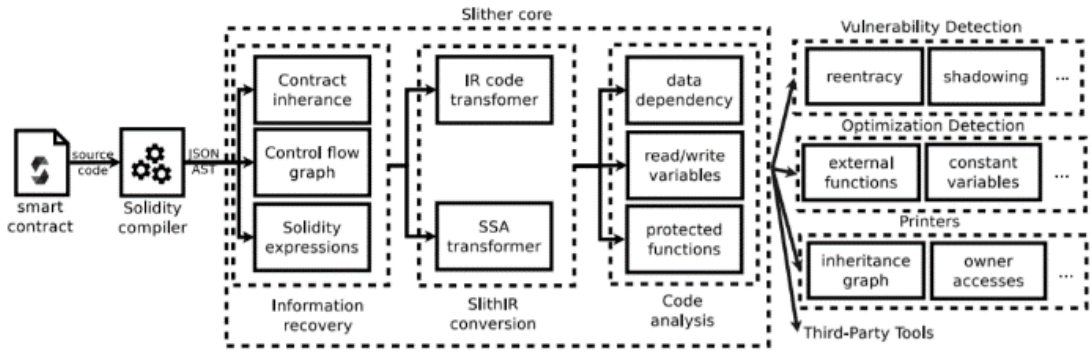

圖 3：Slither 程式架構圖[1]

智能合約 source code 經過 solc 編譯成為 JSON 格式的 Solidity 抽象語法樹 (Abstract Syntax Tree, AST)，再經過 information recovery 可生成合約的繼承圖，進行 IR (Intermediate language) 轉換將 Solidity 抽象語法樹的程式碼轉換為內部表示語言，就可進行 Code analysis，分析合約中 variable、function 的依賴關係，經過以上處理之後，就可以提供檢測漏洞、檢測優化以及理解輸出等功能。

參、方法

3.1 基於 Slither 之檢測工具

本研究以開源檢測工具 Slither 為基準，試著增加許多新的檢測器 (detectors) 來提升 coverage rate。我們根據 Slither 沒偵測到的漏洞 (表 2) 撰寫 detector，以 python 為撰寫語言，並且為每一種漏洞獨立撰寫成一個檔案，以方便單獨檢測某種漏洞。每一個 detector 中，我們會根據漏洞形成的原因寫成規則，再依據規則去做判斷。寫完 detector 之後，將樣本合約一個一個帶入 Slither 以算出 precision rate 以及 recall rate。總計針對 19 個智能合約的安全漏洞設計新的 detector。

我們將我們的 LadyBugs*與 Slither 做比較，表 3 為針對 JiuZhou 新增 detector 的結果，O 為原本 Slither 就可以偵測到的漏洞，⊕代表我們新增的 detector，X 則代表 Slither 和 Ladybugs 都沒有偵測到的漏洞，我們總共增加了 19 個 detector，剩下 3 種漏洞沒辦法被檢測。

表 3：LadyBugs 可偵測資料集 JiuZhou 49 種漏洞中的 46 種

漏洞種類	偵測
1. Unintialized Storage Variables	O
2. Unintialized Local/state Variables	O
3. Wrong Operation	O
4. Integer Sign	⊕
5. Integer Division	O
6. Integer Overflow and Underflow	⊕
7. Integer Truncation	⊕
8. Hidden Built-in Symbols	O
9. Hidden State Variables	O
10. Incorrect Inheritance Order	⊕
11. Right-To-Left-Override Control Character	O
12. Delete Dynamic Array Elements	⊕
13. Using continue-statements in do-while-statements	⊕
14. Re-entrancy Vulnerability	O
15. Unhandled Exception	O
16. Forced To Receive Ether	O
17. Locked Ether	O

* LadyBugs 的連結: https://github.com/Yu-qing/Ladybugs

18. Pre-sent Ether	O
19. Call/delegate call Data/address is Controlled Externally	O
20. Hash Collisions with Multiple Variable Length Arguments	⊕
21. Short Address Attack	⊕
22. Signature With Wrong Parameter	⊕
23. Nonstandard Token Interface	O
24. Returning Results Using Assembly Code in The Constructor	O
25. Specify Function Variable as Any Type	O
26. Transaction Order Dependence	⊕
27. Results Of Contract Execution Affected by Miners(timestamp & random)	O
28. DOS By Non-existent Address or Malicious Contract	⊕
29. DOS By Complex Fallback Function	⊕
30. DOS By Gaslimit	O
31. Storage Overlap Attack	⊕
32. Invariant State Variables Are Not Declared Constant	O
33. Byte[]	⊕
34. Invariants in Loop	⊕
35. Unused Public Functions within The Contracts Should Be Declared External	O
36. Wrong Constructor Name	⊕
37. Use ts.origin For Authentication	O
38. Replay Attack	X
39. Suicide Contract	O
40. Wasteful Contracts	X
41. Non-Public Variables Are Accessed by Public/external Functions	⊕
42. Public Data	⊕
43. Improper Use of Require, Assert, and Revert	X
44. View/constant Function Changes Contract State	O
45. Unlimited Compiler Versions	O
46. Implicit Visibility Level	⊕
47. Nonstandard Naming	O
48. Too Many Digits	O
49. Use Deprecated Built-in Symbols	O

3.2 基於 Ladybugs 之檢測平台

本研究以檢測工具 Ladybugs 為基準，架設一個具有動態掃描區塊鏈上智能合約進行檢測的平台-Contract Pecker，並增加漏洞介紹的功能。

我們以 Ladybugs 做為檢測工具，使用 Python、Flask、javascript、css 程式語言架設平台。平台會不斷抓取並更新最新一批的智能合約地址並紀錄，接著將各個地址的合約內容進行檢測，最後將檢測結果轉為 json 檔儲存。平台的 Latest Contracts 頁面會對最新一批合約檢測結果之 json 檔進行統計，列出各影響程度 (High, Medium, Low, Informational) 的常見漏洞前三名、High, Medium, Low 三者合併之前十名常見漏洞及其數量，讓開發者可以了解在最新一批智能合約中有那些常見的漏洞。

我們還會根據 JiuZhou、Slither、not-so-smart-contract 資料集和 SWC Registry 網站的漏洞資訊編寫漏洞介紹，並實作部分漏洞合約在以太坊測試鏈上發佈，透過與其交互了解漏洞原理，將其彙整為漏洞介紹。內容包括描述、原因、含漏洞合約及修正合約，同時對漏洞合約作註解，並針對漏洞提出防範建議，讓開發者可以根據介紹對漏洞有所了解，再依據建議避免漏洞的發生。

肆、研究成果

本研究成果包含兩個部份：Ladybugs 與 ContractPecker，前者負責後端檢測核心部份，後者負責前端介紹功能部份，分別陳述如下：

4.1 Ladybugs 效能評估

以下表 4 為 Slither 和 LadyBugs 在 JiuZhou 資料集下的 precision rate、recall rate 比較；表 5 為 Slither 和 LadyBugs 在 SmartBugs 資料集下的 recall rate 比較；表 6 為在 LadyBugs 在 SolidiFi 資料集下的 precision rate、 recall rate：

表4：Slither和LadyBugs在JiuZhou dataset下的Precision rate、Recall rate比較

JiuZhou	Slither				LadyBugs			
	TP	FP	Precision rate	Recall rate	TP	FP	Precision rate	Recall rate
Total	36	13	73%	51%	63	13	83%	89%

表5：Slither和LadyBugs在SmartBugs dataset下的 Recall rate比較

SmartBugs Curated	Slither			LadyBugs		
	TP	Bugs	Recall rate	TP	Bugs	Recall rate
Access Control	6	21	29%	6	21	29%
Arithmetic	0	23	0%	23	23	100%
Bad Randomness	3	15	20%	8	15	53%
Denial Of Service	0	8	0%	2	8	25%
Front Running	0	7	0%	5	7	71%
Other	3	3	100%	3	3	100%
Reentrancy	31	31	100%	31	31	100%
Short Addresses	0	3	0%	3	3	100%

表 6：Ladybugs 在 SolidiFi 資料集下的 Precision rate、Recall rate

增加的漏洞檢測	TP	FP	Bugs	Recall rate	Precision rate
Integer overflow/underflow	1451	60	1840	79%	96%
Timestamp dependency	1380	58	1380	100%	96%
TOD	1332	0	1332	100%	100%
Unhandled exceptions	1242	0	1374	90%	100%

可以發現原本 Slither 偵測不到的漏洞 (表 2)，LadyBugs 可以偵測到並且 precision rate 與 recall rate 通常為 100%，可看出 precision rate 從 73%提升至 83%，recall rate 從 51%提升至 87%。平均計算下來，Ladybugs 的 precision rate、recall rate 以及 coverage rate 都比 Slither 明顯高出許多 (圖 4)。

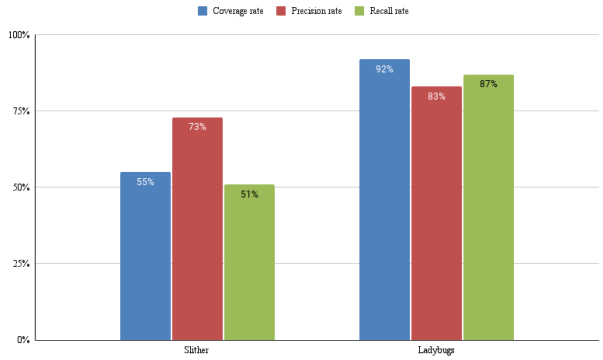

圖 4：Slither 和 LadyBugs 的 Coverage、Precision、Recall rate 比較

在此比較了 LadyBugs 與其他檢測工具之間的執行速度 (表 7)，在與其他工具比較中，Slither 是這九種檢測工具中執行速度最快的，執行一個智能合約只要 0.92 秒，而 LadyBugs

是以 Slither 為基準增加 detector，而越多 detector 會影響到執行時間，但是跟其他工具比較還是相對快非常多的，執行一個智能合約的時間約為 1.2 秒。

我們又將 SmartBugs 資料集帶入 Slither 與 LadyBugs 來比較不同資料量的執行速度(表 8)，SmartBugs 分為兩個資料集，一個為 Curated 資料集共有 143 個智能合約，另一個為 Wild 資料集共有 47,398 個合約而我們只取其 2000 個，撰寫程式碼來自動化測試並計算出檢測時間 (以秒為單位)，可發現在每一個檔案所需秒數只相差了 0.3 至 0.7 秒，在提高 coverage rate 的狀態下不影響效能下降。而 Wild 資料集比 Curated 資料集慢的原因，一部分是因為資料集較大，另一部分原因是因為 Wild 資料集是存放合約地址，須從網路上抓下來才能做分析，而 Curated 是直接在本機執行，速度相對快很多。

表 7：各個檢測工具執行時間比較

Tools	Avg. time	Total time
Honeybadger	46 (s)	0:53:11
Maian	177 (s)	3:23:50
Manticore	491 (s)	5:03:04
Mythril	73 (s)	1:23:42
Osiris	44 (s)	0:50:03
Oyente	36 (s)	0:41:29
Securify	60 (s)	1:09:08
Slither	0.92 (s)	0:02:12
Smartcheck	6 (s)	0:06:34
Ladybugs	1.2 (s)	0:02:52

表 8：Slither 與 LadyBugs 在不同資料量下的執行時間比較

SmartBugs	Curated		Wild	
Tool	Slither	LadyBugs	Slither	LadyBugs
Total time	132.23	172.3	6053.34	7458.21
Total files	143	143	2000	2000
time/per file (s)	0.92	1.20	3.03	3.73

4.2 ContractPecker 平台頁面

平台可以上傳檔案或已上鏈智能合約地址進行檢測，得到檢測結果後可以藉由漏洞介紹了解那些漏洞的成因及影響，並透過範例漏洞合約、修正合約及建議來避免漏洞的出現，同時可以查看以太坊上最新智能合約之檢測結果，了解在不同影響層級下最常使用的 detector、其對應的漏洞及出現次數，系統架構圖如圖 5 所示。

圖5：平台系統架構圖

在 Latest Contracts 頁面可以查看最新一批智能合約檢測結果的統計，會根據影響層度分別排列，可以知道哪些 detector 出現最多次，它所對應的漏洞及出現次數，如圖 6 顯示部分範例。

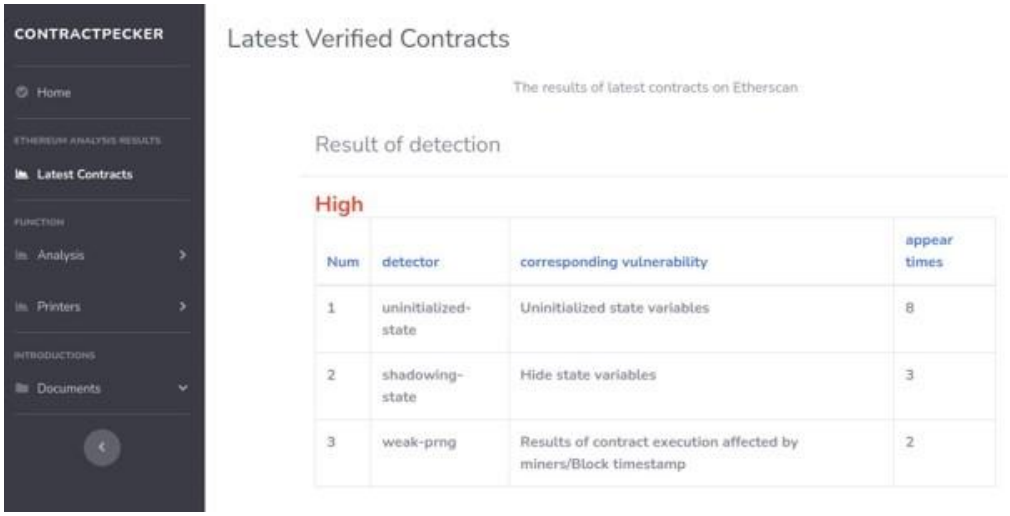

圖 6：最新智能合約檢測結果

平台可以上傳檔案或已上鏈智能合約地址進行檢測 (圖 7)，可選擇 compiler 及區塊鏈，也可以選擇輸出結果的種類，如圖 8 顯示的 human-summary。

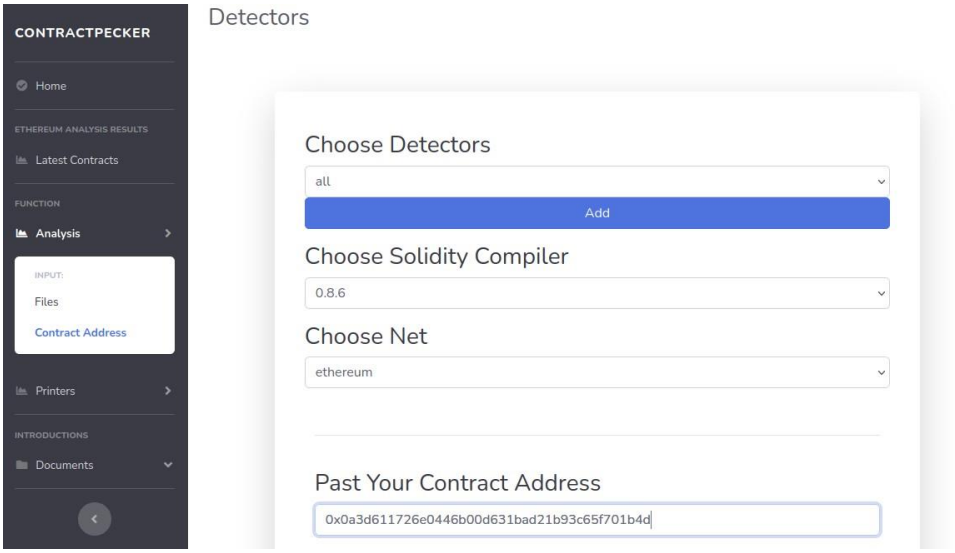

圖 7：上傳智能合約地址進行檢測 (可選擇檔案上傳)

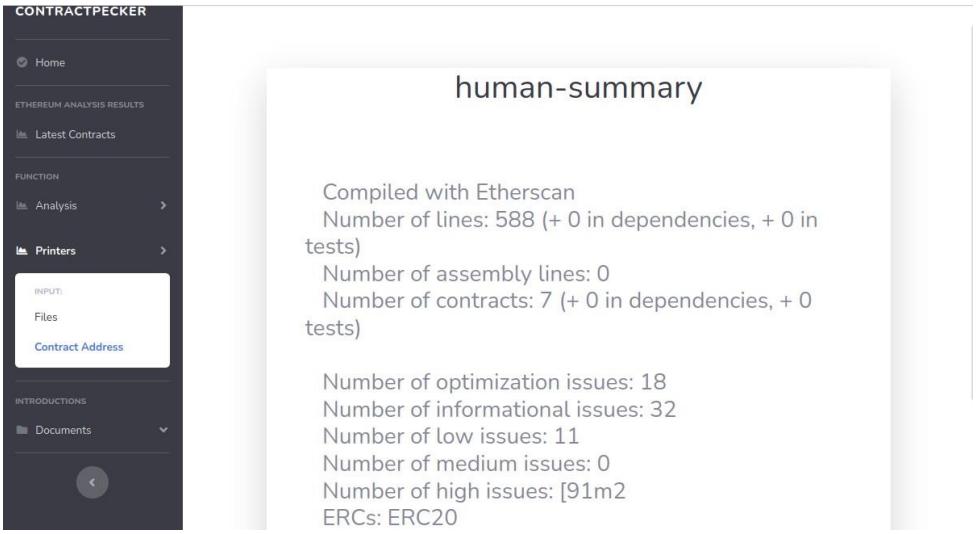

圖 8：選擇 human-summary 的檢測結果

可以在 Documents 列表中查看所有的 detector 與漏洞介紹，圖 9 展示所有漏洞的簡介，點擊漏洞名稱處會出現更詳細的漏洞介紹，包含漏洞名稱、對應 detector、影響層級、描述、原因、建議、範例漏洞合約、程式碼漏洞所在、修正或無漏洞合約、及修正了哪些部分。讓開發者可以初步了解漏洞表現形式並根據建議避免漏洞，如圖 10。

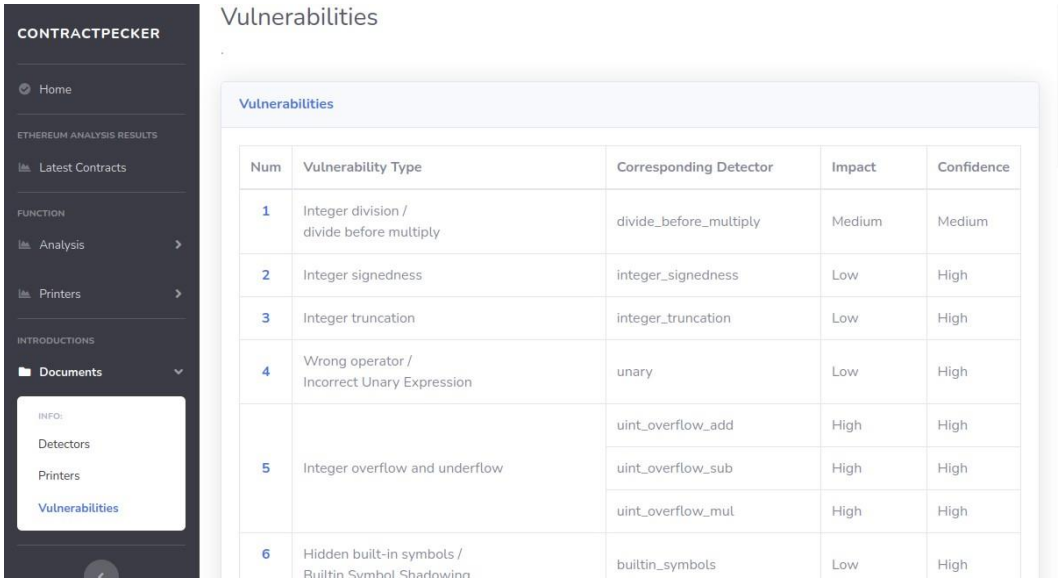

圖 9：漏洞介紹排序

圖 10：漏洞介紹內容頁面

伍、結論

由於區塊鏈去中心化的技術使智能合約無須第三方即可自動執行，因此吸引了不少產業關注，像是非常看重隱私的醫療業、金融業等行業，然而，這項技術越來越受到關注以及應用，也更容易引起攻擊者的注意，使得攻擊手法越來越進步，從而導致一些嚴重的漏洞被惡意利用。根據智能合約的特性以及目前工具的侷限，開發更全面的檢測工具是必要的，本研究將原本 Slither 的 coverage rate 55%改良至 94%，使開發人員更容易審查自己的

智能合約，這也是最主要的貢獻之一。由於大多數漏洞的出現可能是因為開發者對漏洞不夠熟悉，不清楚怎樣的程式碼會存在漏洞，因此在編寫合約時沒有提防漏洞的產生，如果開發者對漏洞有所了解，便可以修改合約內容，盡可能避免漏洞的出現。藉由我們架設的平台，開發者可以透過查看以太坊上最新智能合約之檢測結果，來得知最近的合約中較常出現哪些漏洞，並經由漏洞介紹知曉漏洞的成因及影響，根據平台提供的建議修改合約內容，以盡量避免將有漏洞的合約發佈到區塊鏈上，導致該合約的漏洞受到利用，造成損失。目前 Latest Contracts 頁面需要花費 10 分鐘左右進行抓取及檢測，未來將從多工的方面下手，盡可能減少每次刷新頁面所需的時間。除此之外，未將新增抓取全區塊鏈中智能合約的功能，並以此為基礎建立雲端資料庫，可以利用這些資料分析每年智能合約數量及漏洞數量等的趨勢，還可以了解智能合約每年的種類分布、漏洞種類出現次數等，都是可以進一步探索的內容。

[誌謝]

本研究獲得科技部計畫 MOST 107-2221-E-415 -001 -MY3、110-2221-E-415-006-MY2、110-2813-C-415- 076-E 及 111-2813-C-415 -018 -E 之部份補助。

參考文獻

[1] Böhme, R., Christin, N., Edelman, B., & Moore, T. (2015). Bitcoin: Economics, technology, and governance. *Journal of economic Perspectives*, 29(2), 213-38.

[2] Feist, J., Grieco, G., & Groce, A. (2019). Slither: a static analysis framework for smart contracts. *In 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain* (WETSEB) (pp. 8-15).

[3] Ferreira, J. F., Cruz, P., Durieux, T., & Abreu, R. (2020). SmartBugs: a framework to analyze solidity smart contracts. *In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (pp. 1349-1352).

[4] Ghaleb, A., & Pattabiraman, K. (2020). How effective are smart contract analysis tools? evaluating smart contract static analysis tools using bug injection. *In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 415-427).

[5] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 21260.

[6] Torres, C. F., Schütte, J., & State, R. (2018). Osiris: Hunting for integer bugs in Ethereum smart contracts. *In Proceedings of the 34th Annual Computer Security Applications*

Conference (pp. 664-676).

[7] V. Buterin, "Ethereum white paper," 2013.

[8] Zhang, P., Xiao, F., & Luo, X. (2020, September). A framework and dataset for bugs in Ethereum smart contracts. *In 2020 IEEE International Conference on Software Maintenance and Evolution* (ICSME) (pp. 139-150).

[9] 比特幣市值飆破 5600 億美元擠下台積電 - 數位時代. https://www.bnext.com.tw/article/60820/bitcoin-hack-issue (2023/2/3).

[10] Dapp Ranking. https://www.dapp.com/dapps (2023/2/3).