Communications of the CCISA
Vol. 27 No. 2 May 2021

智慧合約於分散式金融應用之漏洞攻擊解析與解決方案

徐宛萱[1]、林詠章[2]
國立中興大學資訊管理學系
g108029229@smail.nchu.edu.tw
iclin@nchu.edu.tw

摘要

分散式金融於 2020 年後半年開始蓬勃發展，資安事件也相繼爆發，主要多與程式碼安全相關。目前各項分散式金融 (DeFi) 協議的技術尚未成熟，在不同應用層面的潛在風險可能在安全審計時無法被發現，未來更多結合不同協議漏洞的未知攻擊也必然會發生。本論文欲利用目前各類常見 DeFi 應用可能產生之漏洞進行攻擊解析，包含閃電貸、預言機、治理項目等應用，針對 Unstoppable、Naive Receiver、Truster、Side Entrance、The Rewarder、Selfie、Compromised 及 Puppet 等八種可能漏洞進行攻擊解析，進而提供智慧合約安全之撰寫或解決方式，使分散式金融項目佈署於乙太坊後能由源頭之程式碼進行安全控管，從根本減緩來自外部的攻擊。

關鍵詞：智慧合約安全、重入攻擊、區塊鏈安全

Communications of the CCISA
Vol. 27 No. 2 May 2021

Analysis and Solution of Exploiting Vulnerabilities of Smart Contracts in Decentralized Financial Applications

Wan-Shiuan Hsu[1], Iuon-Chang Lin[2]
Department of Management Information Systems, National Chung Hsing University
g108029229@smail.nchu.edu.tw
iclin@nchu.edu.tw

Abstract

Decentralized finance began to flourish after June 2020, and security incidents also broke out one after another, mostly related to code security. At present, the technology of various decentralized finance (DeFi) protocols is not yet mature, and potential risks at different application levels may not be discovered during security audits. In the future, more unknown attacks that combine different protocol vulnerabilities will inevitably occur. This paper intends to use various common DeFi applications such as flash loans, oracles, and governance projects to analyze the following vulnerabilities in a total of eight attack processes: Unstoppable, Naive Receiver, Truster, Side Entrance, The Rewarder, Selfie, Compromised, and Puppet. It also provides smart contract security writing or resolution methods for the analysis and solution of various attack vulnerabilities, so that distributed financial applications can be safely controlled by the source code after they are deployed on Ethereum, and fundamentally slow down external attacks.

Keywords: Smart Contract Security、Reentrancy、Blockchain Security

壹、前言

分散式金融也稱為 DeFi (Decentralized Financial) ，是建立在某些公共區塊鏈之上的金融應用生態系統[1]，旨在使傳統金融生態系統中的所有金融工具 (如期權，期貨，衍生產品等) 以及所有金融服務 (如貸款，借款，轉移等) 在分散的世界中得以實現。分散式金融之優點為對所有人開放[2]，可以將更多服務推廣給大眾、不受國籍限制、更加透明、金融產品選擇多樣。以股市為例，交易受時間限制，分散式金融 24 小時皆可進行交易。以借貸為例，在銀行進行借貸需要相當多道手續及相關證明文件，分散式金融不需經過審核，藉由抵押資產即可進行借貸，使得隨時隨地任何人皆可以享受金融服務而不受門檻所限。分散式金融項目近日急速成長，於 2020 年 9 月鎖倉價值超過一百億美元[3]。分散式金融之缺點為目前有法律監管相關議題[4]、乙太坊近日手續費偏高[5]，以 Uniswap 為例，手續費抽近三成[6]，另外由於法律監管問題及鎖倉價值相當高等誘因，致使資安事件頻傳。區塊鏈技術具有不可竄改之特性，智慧合約於乙太坊佈署後若受到攻擊無法馬上作出反饋，每次安全事件所造成之損失約數十至百萬美元。

法律相關議題目前不可控，手續費部分將會在乙太坊 2.0 擴容後有相關配套措施[7]，因此針對資安事件部分進行探討。第一起且最為知名的智慧合約攻擊事件便是 2016 年所發生的「The DAO」事件[8]，駭客藉由智慧合約之重入漏洞盜領了約 370 萬個乙太幣，約價值約 7,200 萬美元。此次攻擊事件也造成了乙太坊硬分岔，區塊回復至發生攻擊前，編號 1,920,000 的區塊重新開始計算。

2020 年所爆發之 DeFi 安全事件多與智慧合約撰寫有關，其中閃電貸攻擊及利用重入 (Reentrancy) 漏洞進行跨合約攻擊佔了多數，以 dForce 受到 ERC777 重入攻擊為例[9]，Lendf.Me 與 imBTC 兩個合約本身程式碼無安全問題，而兩者組合產生的協議帶來系統性的風險，此類可組合性[1]對 DeFi 的發展有極大的安全隱患。由於並無法全面對所調用之外部合約進行檢測，在合約檢測的方面[10]，現有的工具有 MythX、Mythril、Slither、 Contract-Library、 Echidna、 Manticore、 Oyente、 Securify、 SmartCheck、 Octopus、 sFuzz、 Vertigo 等等，這些工具檢測的方式各有優缺，針對跨合約函數調用相關的漏洞有其侷限性。因此研究如何應對相關攻擊為目標，藉由解析攻擊手法及統整智慧合約方面之漏洞，提出解決方法與建議，使得在開發過程中除了以往利用智慧合約檢測工具、全面之安全審計、保險方式等等以外，可以更安全之撰寫方式保護智慧合約，從根本避免外部調用所引起之漏洞，使分散式金融產品於佈署後可以將遭到攻擊之風險降至最低。

貳、文獻探討

本節介紹於智慧合約漏洞中最常見之重入漏洞，以及於 2020 後半年出現之 DeFi 新

型態攻擊。

2.1 重入 (Reentrancy) 漏洞

隨著智慧合約發展，藉由重入漏洞進行攻擊的方式也漸趨多樣，將其分為三種態樣[11]，分別為跨函數重入 (cross-function re-entrancy) 、委託重入 (delegated re-entrancy) 及基於創建的重入 (create-based re-entrancy) :

1. 跨函數重入 (cross-function re-entrancy)

攻擊者能夠透過兩個不同功能但共享同一狀態的函數實現跨函數攻擊。以圖 1 進行說明，withdrawBalance () 與 transfer () 共享 userBalances 狀態，withdrawBalance () 在執行 call.value () 後將用戶的餘額歸零，但在進行 call.value () 後才將用戶餘額歸零，有可能發生在調用 call.value () 成功後，清空用戶餘額前，被重入呼叫 transfer () ，transfer () 並沒有將用戶餘額清除，因此即使攻擊者已收到款項仍然可以不斷轉移金額。知名的 The DAO[8]即是利用此種方式將合約內的金額進行轉移。現有的合約檢測工具無法針對此類型漏洞精確進行偵測及分析。

```
//Cross-function Reentrancy
contract INSECURE{
mapping (address => uint) private userBalances;

function transfer(address to, uint amount)public{
    if (userBalances[msg.sender] >= amount) {
       userBalances[to] += amount;
       userBalances[msg.sender] -= amount;
    }
}

function withdrawBalance() public {
    uint amountToWithdraw = userBalances[msg.sender];
    (bool success, ) = msg.sender.call.value(amountToWithdraw)("");
    require(success);
    userBalances[msg.sender] = 0;
}}
```

圖 1、跨函數重入範例

2. 委託重入 (delegated re-entrancy)

允許一個合約在當前合約的上下文環境中調用其它合約，攻擊者可能藉此利用漏洞實現委託重入攻擊。靜態分析工具在進行離線分析時，無法確知合約在實際執行時會委託調用的合約，故無法檢測委託重入攻擊。

3. 基於創建的重入 (create-based re-entrancy)

與 Delegated reentrancy attack 相似，合約的 constructor 被攻擊者掌握或控制，現有分析工具在進行離線分析時，無法確知合約在實際創建時會調用的合約，故無法檢測基於創建的重入攻擊。

2.2 DeFi 新型態攻擊

DeFi 於 2020 年後半年開始蓬勃發展，駭客攻擊事件頻傳，除了鎖倉價值相當高及法律目前無法完全約束以外，主因在於推陳出新的項目技術尚未成熟，各協議間的可組合性[1]使得漏洞難以被完整檢測。

1. 閃電貸攻擊

以 2020 年 2 月 15 日，第一起受到閃電貸攻擊之 bzx 借貸協議的攻擊過程進行說明[12]：

(1) 攻擊者首先透過 bzx 協議借貸 10,000 個 ETH，並將 5,500ETH 存入 Compound 作為抵押品，並貸出 112 WBTC。

(2) 攻擊者於 bzx 流動池存入 1,300 ETH，並調用 bzx 槓桿交易功能開五倍槓桿，拿到 5,637.62 個 ETH。

(3) 攻擊者透過 KyberSwap 將 5637.62 個 ETH 兌換成 51.345576 WBTC，此處做空 ETH 是借來的 5 倍。本次交易導致將 WETH / WBTC 的兌換率提高到 109.8 ，大約是正常兌換率 (～38.5 WETH / WBTC) 的 3 倍，KyberSwap 基本上會查詢其儲備金並找到最優惠的匯率，最終只有 Uniswap 能提供這樣的流通性，因此這個交易從本質上推動了 Uniswap 中 WBTC 價格上漲了 3 倍。

(4) 攻擊者從 Compound 借出來的 112 WBTC 在 Uniswap WBTC pool 賣 112 WBTC，獲得 6,800 ETH。

(5) 攻擊者歸還借貸的 10,000 個 ETH。整起攻擊事件發生時間大約 13 秒，攻擊者大約獲得 1,271 個 ETH 收益 (約三十五萬美元) ，此類型攻擊多是跨合約，並且結合金融操作 (貸款、操作槓桿、拉抬價格等) 。交易紀錄如圖 2：

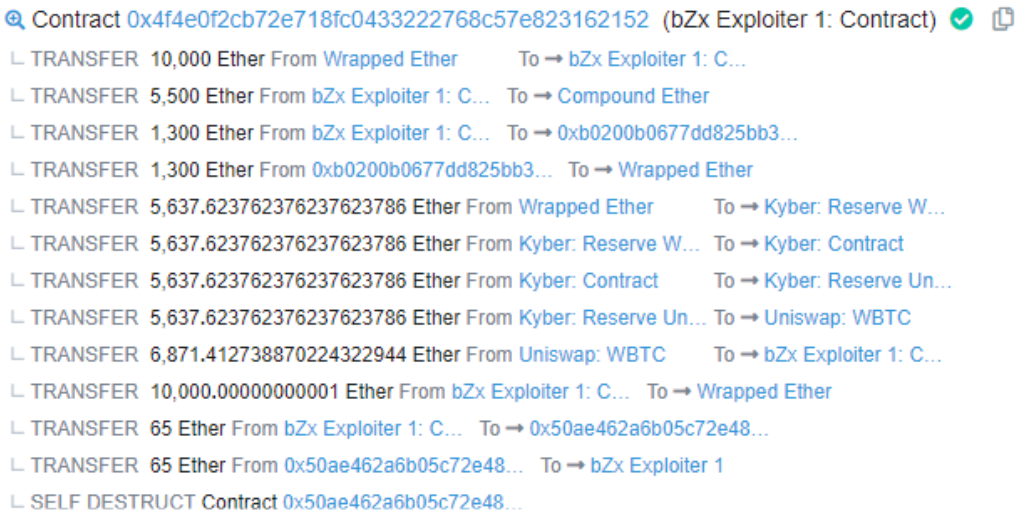

圖 2、bzx 協議受閃電貸攻擊之交易紀錄

2. ERC777 重入 (Reentrancy) 攻擊

於 2020 年 4 月 18 日，Uniswap 及 Lendf.Me 分別受到重入攻擊，與過往的重入攻擊較為不同之處在於利用 ERC777 代幣標準的特性進行重入。使用 ERC777 代幣標準的代幣會在每一次發生代幣轉帳時去調用代幣發送者的 tokensToSend 函數。以第一起遭到 ERC777 重入的 Uniswap[13]說明如何利用 ERC777 特性進行攻擊:

(1) 駭客先向 Uniswap 合約要求一些 imBTC，並將 imBTC 換成 ETH。
(2) Uniswap 會先將 imBTC 發送再換成 ETH，由於 imBTC 使用 ERC777 標準，轉帳後會調用駭客的 tokenToSend () 。
(3) 駭客在 tokenToSend () 進行重入攻擊，在 Uniswap 扣掉駭客的 imBTC 前，駭客再次調用 Uniswap 的 transferFrom () ，不斷用等量的 imBTC 換 ETH。

參、攻擊漏洞解析

本論文完成 Damn Vulnerable DeFi[14] 提出的八個案例之漏洞進行攻擊解析，提供可能的解決方案以符合 SCSVS V14:Defi 所訂立之安全檢核標準。開始運行前需要設定環境，每個案例皆有攻擊達成條件，所有案例皆需要使用攻擊者帳號，有些案例需要自行撰寫攻擊者合約。藉由達成攻擊條件並提出符合 SCSVS 檢查清單之可能之解決方法。透過 VSCode 運行環境，使用 JavaScript 撰寫智慧合約後端程式碼，使用 Remix IDE 測試合約間互動、撰寫攻擊者合約及防禦合約。

3.1 Unstoppable

Unstoppable 提供一個可以快速借貸的借貸池，達成攻擊的條件為使資金被鎖在池內，無法再提供借貸服務。案例含有三個合約，分別是以 ERC20 標準所發行的代幣合約、提供服務的借貸合約以及與借貸合約互動的使用者合約。合約運作及攻擊流程如圖 3：

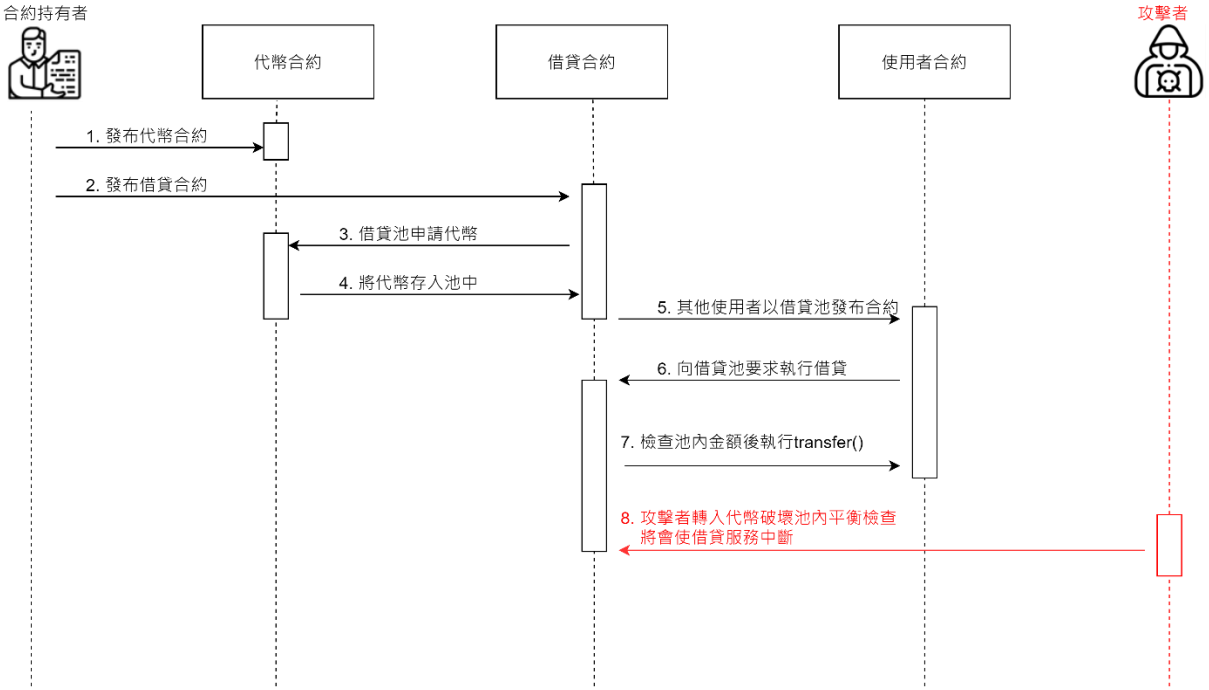

圖 3、Unstoppable 流程圖

3.2 Naive Receiver

Naive Receiver 含有兩個合約，分別是一個提供閃電貸的借貸池合約，以及一個與借貸池掛鉤提供池借貸金額的使用者合約。達成條件為將其他提供借貸的使用者合約內金額掏空。合約運作及攻擊流程如圖 4：

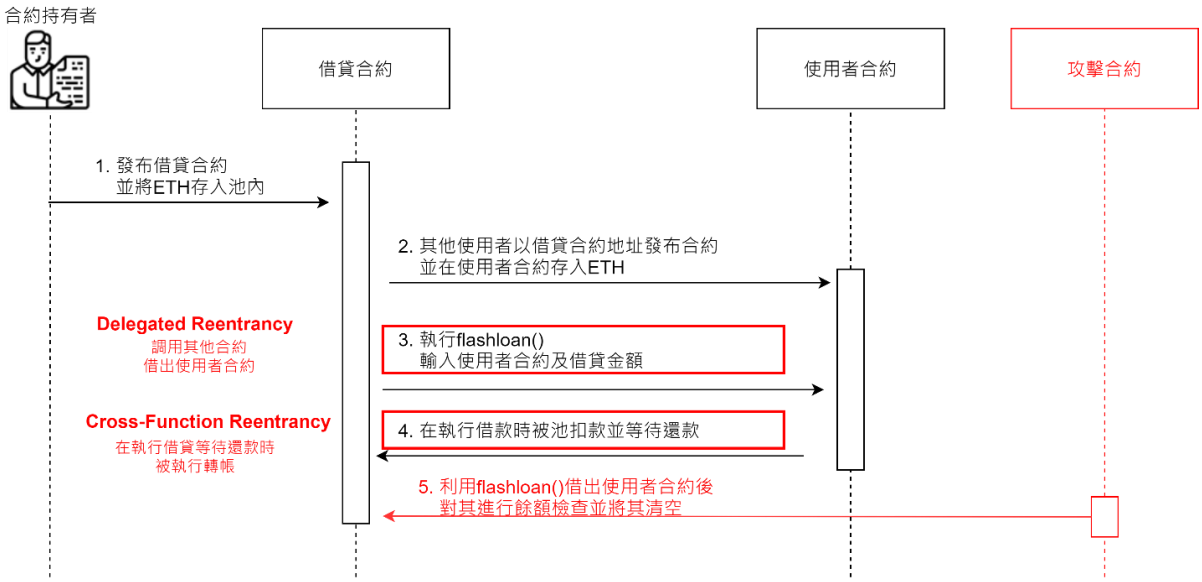

圖 4、Naive Receiver 流程圖

3.3 Truster

Truster 含有兩個合約，分別是以 ERC20 代幣標準發行的代幣合約，及提供閃電貸服務的借貸池合約，池內有 100 萬代幣，達成條件為掏空借貸池內的代幣。借貸池合約有做避免重入的保護，其中的 flashloan () 只要金額在同一筆交易內可以還借貸之金額，就提供任何金額的借貸。執行借貸時可以傳遞四個參數，其中兩個參數 (target 和 data) 可以指定合約和函數 (借用代幣時池將會調用它們) ，可以使貸方合約調用任何合約的任何功能。合約運作及攻擊流程如圖 5:

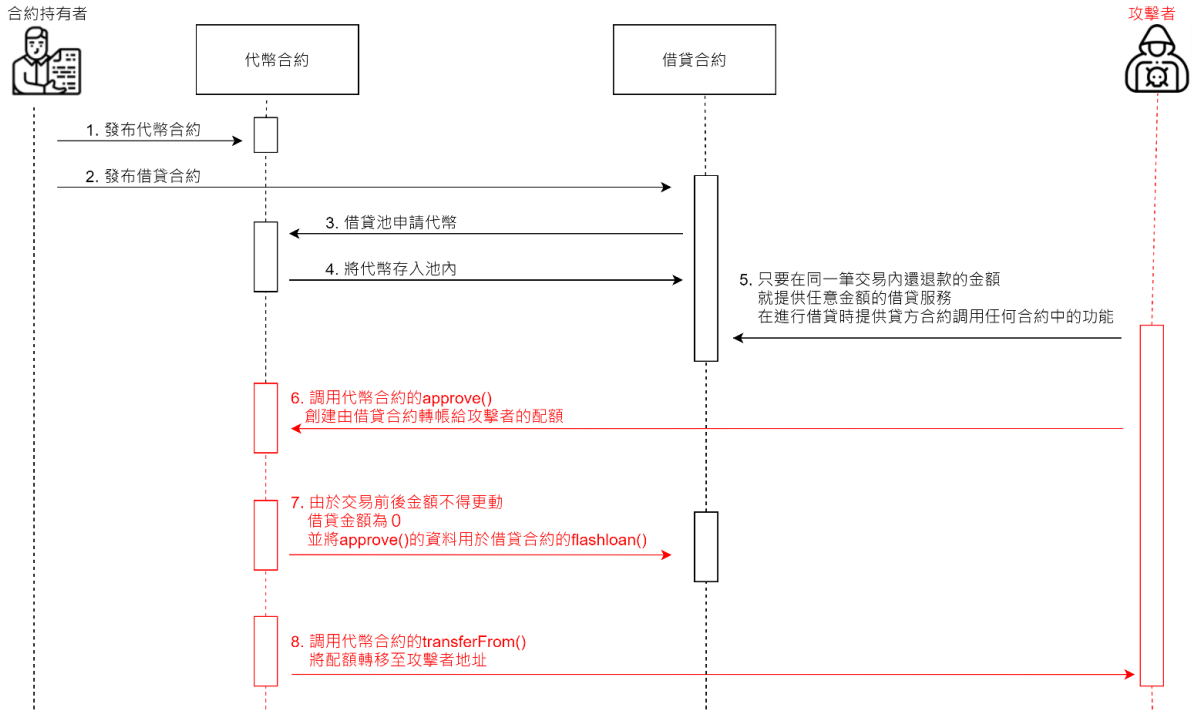

圖 5、Truster 流程圖

3.4 Side Entrance

含有一個合約，是一個原有 1000ETH 的借貸池合約，提供存款及提款，並以合約內的金額提供閃電貸，促進合約內金額流動性，有一個 IFlashLoanEtherReceiver 的 interface，允許使用閃電貸的合約利用借貸的金額去做更有效的運用，借貸池的 flashloan () 當合約內的總金額不變或是更高後就允許執行 execute () ，讓其他合約對借貸池合約內金額進行其他用途，使借出的資金可以更有效的運用。達成條件為清空借貸池內的金額。實驗過程於借貸池內添加餘額查詢函數，以利追蹤合約內 ETH 變化。合約運作及攻擊流程如圖 6：

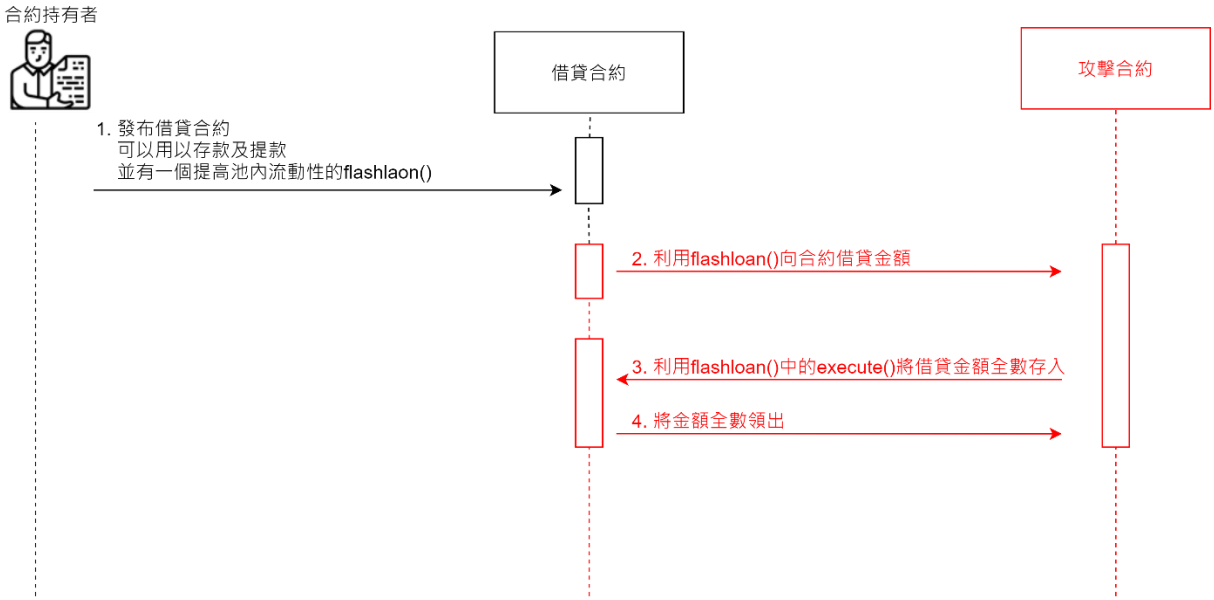

圖 6、Side Entrance 流程圖

3.5 The Rewarder

含有五個合約，分別為以 ERC20 代幣標準所發行的代幣合約、提供代幣閃電貸服務的借貸合約、使用者可以將代幣存入以賺取獎勵代幣的獎勵池合約、以 ERC20 代幣標準所發行作為將代幣存入獎勵池報酬的獎勵代幣合約，以及用以計算使用者將代幣存入獎勵池所應獲得多少報酬的獎勵代幣計算合約。達成條件為將所有獎勵據為己有。獎勵代幣計算合約每五天進行一次快照，記錄將代幣存入獎勵池中的使用者所應獲得的獎勵代幣，獎勵代幣計算根據使用者存入的代幣與已存代幣總數的百分比，使用者可以利用獎勵池合約中的 distributedRewards () 領取獎勵代幣。合約運作及攻擊流程如圖 7：

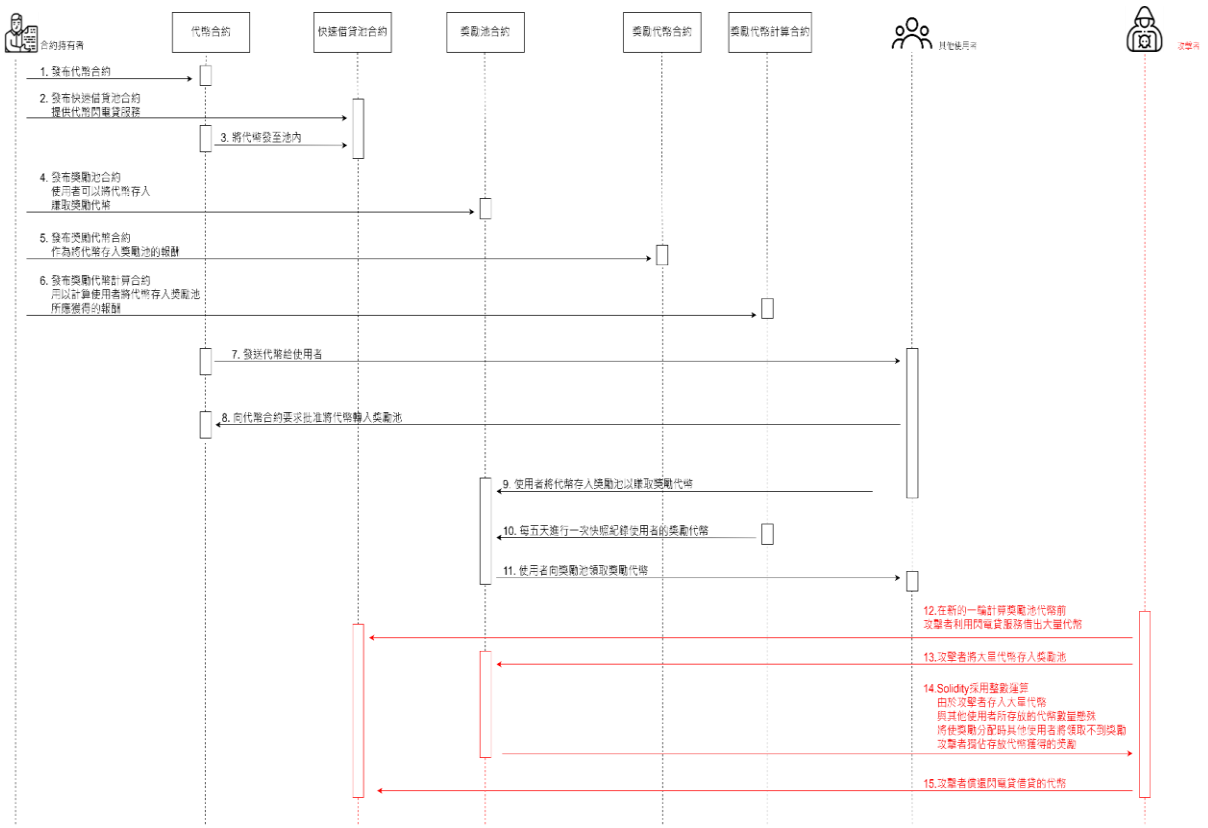

圖 7、The Rewarder 流程圖

3.6 Selfie

含有三個合約，分別為以 ERC20 代幣標準所發行的代幣合約、提供閃電貸並受到治理合約保護的借貸池合約，以及設計治理機制的治理合約。借貸池中有 150 萬個代幣。達成條件為清空借貸池內代幣。借貸池合約 drainAllFunds () 允許所有代幣從池轉移，但受到治理合約的保護。治理合約檢查參與者是否持有足夠的票數，藉由 _hasEnoughVotes () 檢查提案者是否持有超過一半的治理代幣，通過檢查的提案者可以排隊等待並透過治理合約調用函數。合約運作及攻擊流程如圖 8：

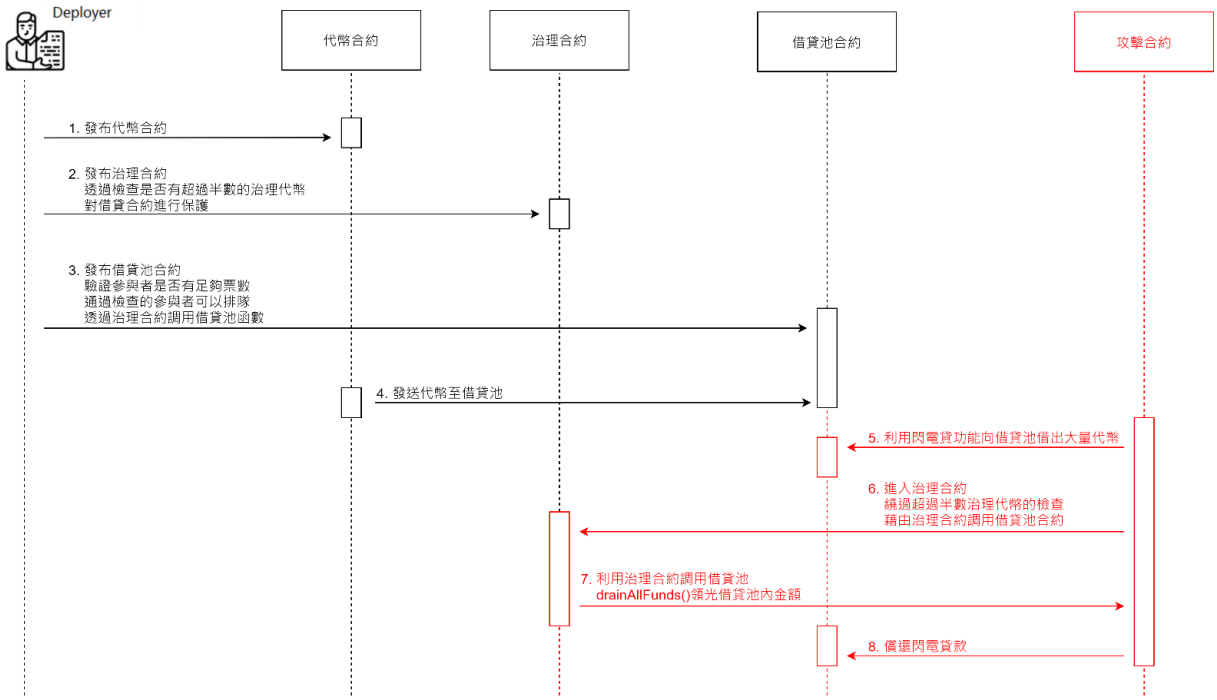

圖 8、Selfie 流程圖

3.7 Compromised

含有四個合約，分別是以 ERC721 代幣標準發行的 NFT 代幣合約、用以購買賣出 NFT 的交易合約、設定可信任的預言機來源的預言機初始化合約以及計算由多個來源預言機價格中位數的報價合約。達成條件為清空交易所內 ETH。網站的回應片段皆是介於 0x20 和 0x7e ASCII code 的範圍內，分別先由 16 進制轉換成 ASCII code，得到以下字串:

MHhjNjc4ZWYxYWE0NTZkYTY1YzZmYzU4NjFkNDQ4OTJjZGZhYzBjNmM4YzI1NjBiZjBjOWZiY2RhZTJmNDczNWE5

MHgyMDgyNDJjNDBhY2RmYTllZDg4OWU2ODVjMjM1NDdhY2JlZDliZWZjNjAzNzFlOTg3NWZiY2Q3MzYzNDBiYjQ4

再利用 base64 進行解碼，分別得到以下字串:

0xc678ef1aa456da65c6fc5861d44892cdfac0c6c8c2560bf0c9fbcdae2f4735a9

0x208242c40acdfa9ed889e685c23547acbed9befc60371e9875fbcd736340bb48

將以上字串以私鑰方式匯入，將會分別對應其中兩個預言機來源帳戶地址。在此案例中僅有設置三個可信任的預言機來源，並且由各個來源自行使用 postPrice () 報價，掌握其中兩個來源帳號，便可以對 DVNFT 代幣價格進行操作。合約運作及攻擊流程如圖 9：

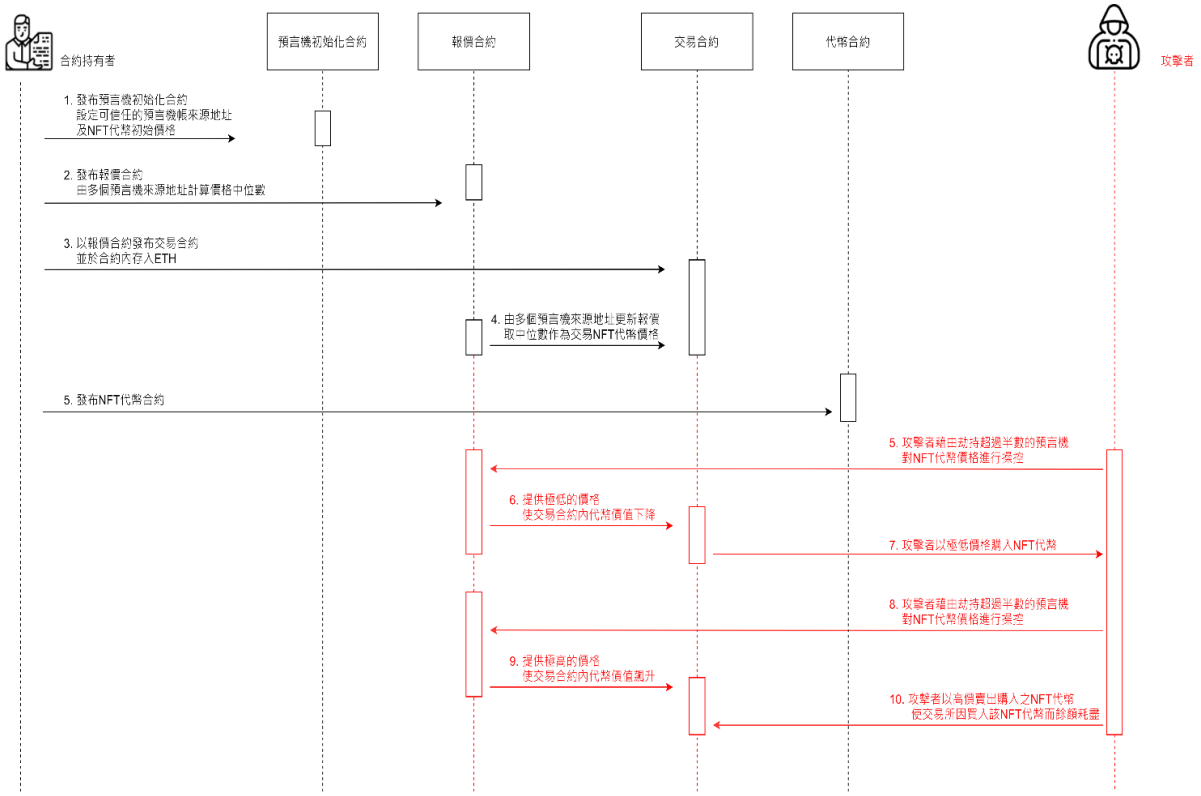

圖 9、Compromised 流程圖

3.8 Puppet

含有五個合約，分別為以 ERC20 代幣標準發行的代幣合約、提供代幣借貸的借貸池合約以及 Un0iswap v1 合約，其中交換模板合約用以為工廠合約進行初始化，工廠合約用以為代幣創建新的交換合約，交換合約內有 ETH 及代幣提供流動性。借貸池的 borrow () 並非閃電貸，對代幣進行借貸需存入兩倍的 ETH 作為抵押。借貸池中代幣價格透過 computeOraclePrice () 計算交換合約內 ETH 與代幣的比率。達成條件為在不損失 ETH 的情況下盡可能借出借貸池內的代幣。合約運作及攻擊流程如圖 10：

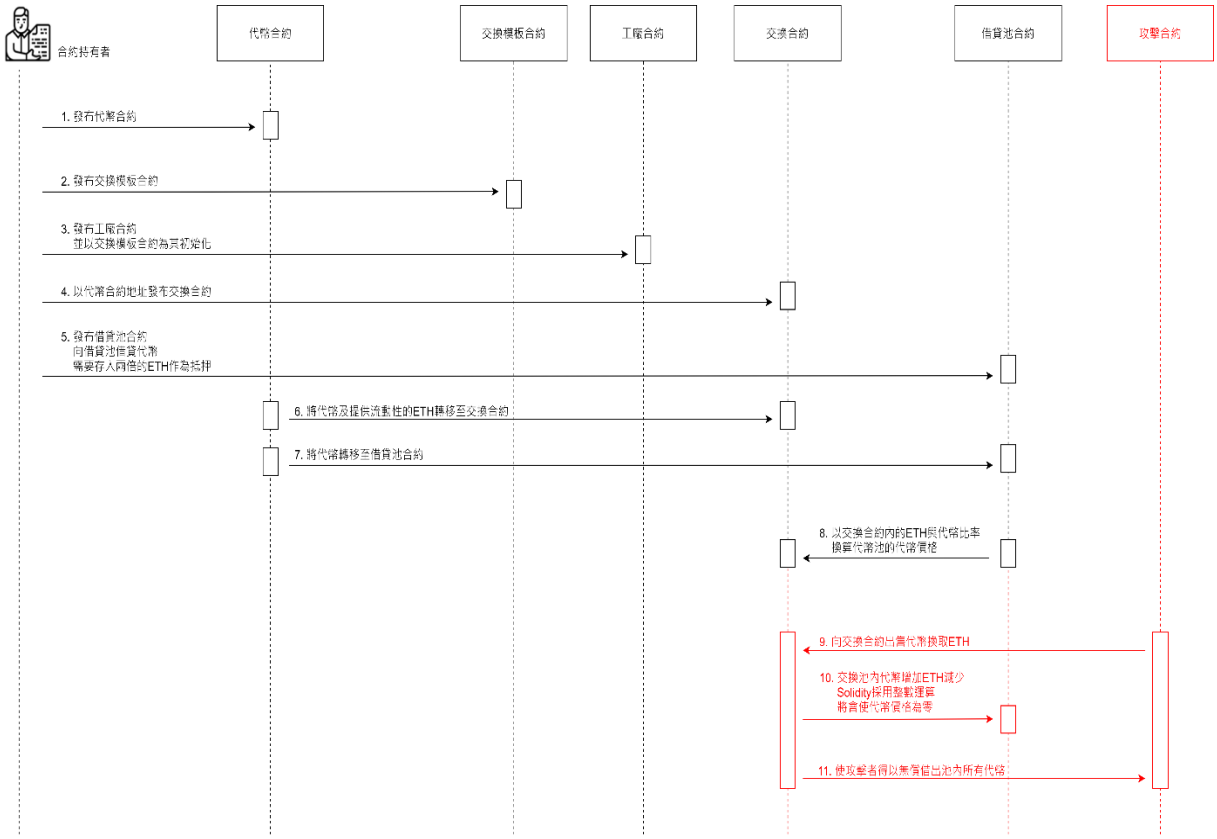

圖 10、Puppet 流程圖

肆、解決方案

4.1 Unstoppable

預防方式為將平衡檢查式修正，對轉入的代幣進行檢查。此案例需要驗證之安全項目為 14.1，合約餘額僅能透過本身進行更改。assert () 多置於函數結尾處，多用以檢查是否有上溢或者是下溢，在完成變化後檢查狀態，避免本不應該發生的情況出現，與 require () 的差異在於 require () 可以退回使用的 gas，assert () 會拿走所有 GasLimit 的手續費。

4.2 Naive Receiver

預防方式為建立借貸池時需要避免從借貸池合約調用其他合約的任何函數，可以在合約內指定需要調用的合約或函數，僅能由受信任的來源或只有借貸池持有者可以調用。此案例需要驗證之安全項目為 14.4，檢查代幣接收方函數的權限控款。需將借貸池上下

文調用其他合約 receiveEther () 部分進行修改，將其修改為在合約內部呼叫，並在使用者合約初次與借貸池進行互動時紀錄使用者合約持有者，限制僅有使用者及借貸池合約持有者可以對使用者合約金額進行扣款。

4.3 Truster

預防方式為限定借貸池可以被調用的合約或函數，須避免任何人可以在池調用任何合約或函數。此案例需要驗證之安全項目為 14.3，檢查閃電貸調用接收方合約預定義的函數。此案例類似 Naive Receiver，差別在於 Truster 受到攻擊的為借貸池。此案例安全的寫法可以 modifier 建立可以對借貸池的 flashloan () 調用的子集，使用 modifier 而非 require 的原因在於若要去定義哪些合約或函數有存取權限時較好管理。

```
function flashLoan(
    uint256 borrowAmount,
    address borrower,
    address target,
    bytes calldata data
)
    onlyowner external
    nonReentrant
{
    uint256 balanceBefore = damnValuableToken.balanceOf(address(this));
    require(balanceBefore >= borrowAmount, "Not enough tokens in pool");
```

圖 11、限制可以調用之函數子集

4.4 Side Entrance

預防方式為借貸合約的 withdraw () 跟 deposit () 需要做防止重入的保護，另外 flashloan () 僅對金額進行檢查，與 withdraw () 與 deposit () 共享 balances[msg.sender]狀態，需要避免針對跨功能共享同一狀態變數的重入攻擊。此案例需要驗證之安全項目為 14.2，檢查借用函數是否不可重入，可以避免在執行閃電貸時更新借貸方餘額的攻擊。安全的寫法需要於借貸合約的 withdraw () 、deposit () 及 flashloan () 使用 mutex，flashloan () 於借貸者執行完 execute () 後將其持有金額清空。

```
modifier lock {
require(!lockBalances,"cannot reentrancy");
lockBalances = true;
_;
lockBalances = false;
}
```

圖 12、mutex

4.5 The Rewarder

預防方式為若代幣的小數位數與 ETH_18 相同，需要提高小數點精確度 (使用 wei 作為單位) 。此案例需要驗證之安全項目為 14.5，檢查流動池內配額是否提高精度進行計算。14.11，有運用到算數之合約需檢查數學運算順序是否正確。14.6，檢查存入代幣及配發獎勵函數不可重入，以避免代幣價值大幅波動。

4.6 Selfie

預防方式為可以於執行提案前設置延遲時間，使擁有投票權的使用者可以對惡意提案有時間做出回應。此案例需要驗證之安全項目為 14.7，檢查合約受治理合約保護時是否可以避免來自閃電貸之攻擊。14.10，檢查是否可以對即便受到信任之合約，於執行過程做出回應 (例如對攻擊行為反饋) 。在治理項目攻擊的可行性方面，MakerDAO 在收到負責任的披露 (Responsible disclosure) 之前[1]，對於治理項目的延遲為零，攻擊者可以透過眾籌或閃電貸累積到治理攻擊所需的資金。在 MakerDAO 進行修復前，只要攻擊者能夠鎖定 2750 萬美元的治理代幣，便能夠竊取在兩個區塊內所有的抵押品，價值約 5 億美元。MakerDAO[15]的治理流程包括提案及執行投票，提案是為了先在社群達成共識，投票的目的是為了批准或駁回對系統內風險參數的更改，其風險參數包含:債務上限、穩定費、清算率、清算罰金、擔保物拍賣期限、最低加價幅度 (拍賣中新競價的最低增幅) 。在 MakerDAO 的設計中，當碰到治理失當、程序漏洞、抵押品價格暴跌等情形，有投票權的使用者可以通過在緊急關停模組 (ESM) 存入代幣觸發緊急關停，達到一定票數關停會立即生效，緊急關停分為三個階段:使用者將取回存放於內的所有資產、抵押品將進行拍賣以及 Dai 持有者贖回剩餘的抵押品。

```
function drainAllFunds(address receiver) external onlyGovernance {
    uint a=block.timestamp;
    require(a>deadline,"cannot execute");
    uint256 amount = token.balanceOf(address(this));
    token.transfer(receiver, amount);

    emit FundsDrained(receiver, amount);

}
```

圖 13、執行提案前設置延遲時間

4.7 Compromised

預防方式為於更新價格限制閥值，若超過閥值則採用舊價格。此案例需要驗證之安全項目為 14.9，檢查更改合約屬性之外部合約是否設定更改之閥值及其他更新之限制。

14.8，檢查當預言機受到攻擊時是否能即時作出回應。14.10，檢查是否可以對即便受到信任之合約，於執行過程做出回應 (例如對攻擊行為反饋) 。對於預言機安全各 Defi 項目各有不同保護機制。

Uniswap V1 預言機曾發生遭利用閃電貸操縱代幣價格的攻擊，由於其更新價格是及時的，在報價設計上並不安全，V2[16]設計時間加權平均價格 (TWAP) 累加器，累加器在每個區塊最後追蹤代幣對累計時間的價格，在兩個時間點對累加器進行採樣，求出差值，然後除以經過的時間，可以得出該時間間隔內代幣對應該區塊最後的價格的時間加權平均價格。比起即時更新價格有兩個關鍵的區別，一為平均價格取決於過去出現過之價格，與時間成正比，因此攻擊者若想操縱價格，就需要使價格維持更長的時間，二為平均價格在區塊中變化並不會受到影響，僅在區塊最後才會，攻擊者必須在區塊的最後對價格維持操縱才能影響平均價格，將會影響攻擊者的攻擊成本。此設計對於價格操縱有相當高的防禦力，適合用於保護鏈上已經存在的流動代幣資產。

MakerDAO 系統的參考價格[17]是透過一個預言機所提供，該預言機可以對來自多個外部價格數據進行整理，外部價格提供者會持續監控多個外部來源的參考價格，並將在以下情況下向區塊鏈提交更新：原始價格與最近提交的價格相差超過定義的數量 (當前為 1%) 、上次價格更新時間超過 6 小時。依據其白皮書[15]，為保護系統設置了價格靈敏度參數，舉例說明:若價格靈敏度參數設置十五分鐘內變化在 5%以內，意指預言機在餵價時價格不能在十五分鐘內有超過 5%的浮動。為避免攻擊者企圖控制多數預言機，採用預言機安全模型 (OSM) 接收價格，而非直接從預言機取得價格，OSM 將價格延遲一小時，若預言機遭到破壞時，緊急用的預言機或是有治理權限的用戶可以投票凍結預言機。OSM 是為了確保在經過指定延遲之前，系統不會使用由預言機所提供的新報價，延遲機制設計的目的是為了確保有時間檢測或對預言機攻擊採取防禦措施。

bzx 考量安全性使用 ChainLink 作為預言機解決方案[18]，價格每有 1%的偏差就會發生更新 (對於 KNC / ETH 和 BTC / ETH，價格為 2%) ，如果未達到偏差閾值，則每小時都會進行最小的基於時間的更新。Aave 使用 Chainlink 價格參考數據合約取得資產價格，價格每有 1%的偏差就會進行更新 (對於較少使用的資產則為 2%) ，如果未達到偏差閾值，則每小時進行一次最少的基於時間的更新。Compound 價格來自 Coinbase Pro、Bittrex、Poloniex 和 Binance 的取乙太幣價格中位數，並在資產中位數價格出現 1%偏差後在鏈上發布，為了安全起見，價格變化 (在協議級別) 被限制為每小時 10%。DDEX 使用鏈上價格預言機，預言機在不同市場的穩定性有所差異，價格確定過程內基於有設計安全機制，以防止預言機受到攻擊，以 USDC 為例，參考 Coinbase 的價格，並在其鏈上代理預言機中內置安全檢查，檢測異常值和到期時間，以 DAI 為例，其價格區間受到限制，目前設置為 0.95 和 1.05。dYdX 針對不同資產使用不同價格預言機。

4.8 Puppet

預防方式為在計算代幣價格時，分子和分母乘以儲備金。此案例需要驗證之安全項目為 14.11，有運用到算數之合約需檢查數學運算順序是否正確。14.12，檢查計算轉換價格時，分子和分母乘以儲備金。在計算借貸代幣所需 ETH 取自交換合約，分子乘以輸出儲備 (此例為交換合約中 ETH 餘額) ，分母應乘以輸入儲備 (此例為交換合約中代幣餘額) 。

伍、結論

本論文透過分析目前常見之分散式金融項目應用案例，利用漏洞實作攻擊並解析手法，藉由統整現行分散式金融項目於各項安全考量之制度設計，針對每一案例提出可能之預防方式，從源頭之程式碼進行安全控管，並使各項案例符合 SCSVS V14:DeFi 安全標準。能夠使分散式金融項目於開發過程更加安全，就算面臨未知之安全事件，上線後可以更快停損，將風險降至最低。

由於分散式金融項目多元，鎖倉資產類型亦五花八門，與傳統金融項目不同之處在於並沒有第三方對價格控管，跨合約金融協議使攻擊者可以對池中價格進行操縱，不同市場及資產間預言機餵價制度設計也會是未來需要考量之安全議題。

參考文獻

[1] L. Gudgeon, D. Perez, D. Harz, B. Livshits, and A. Gervais, "The decentralized financial crisis," in *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)* , 2020, pp. 1-15: IEEE.

[2] Y. Chen and C. J. J. o. B. V. I. Bellavitis, "Blockchain disruption and decentralized finance: The rise of decentralized business models," *ScienceDirect,* vol. 13, p. e00151, 2020.

[3] *Defi Pulse 網站*. Available: https://defipulse.com/

[4] *Salami, Iwa. "Decentralised Finance: The Case for a Holistic Approach to Regulating the Crypto Industry." Salami, I. (2020) 'Decentralised Finance: The Case for a Holistic Approach to Regulating the Crypto Industry'Journal of International Banking and Financial Law 35.7 (2020) : 496-499. [5] Bitcoin, Ethereum Avg. Transaction Fee historical chart. Available: https://bitinfocharts.com/comparison/transactionfees-btc-eth.html#6m*

[6] H. Adams, N. Zinsmeister, and D. J. U. h. u. o. w. p. Robinson. (2020) . *Uniswap v2 core*. Available: https://uniswap. org/whitepaper. pdf

[7] *gas fee*. Available: https://blog.makerdao.com/how-ethereum-2-0-will-address-gas-issues-and-enable-dai-and-defi-to-scale/

[8] *The DAO*. Available: https://en.wikipedia.org/wiki/The_DAO_(organization)

[9] *How the dForce hacker used reentrancy to steal 25 million*. Available: https://quantstamp.com/blog/how-the-dforce-hacker-used-reentrancy-to-steal-25-million

[10] S. Sayeed, H. Marco-Gisbert, and T. J. I. A. Caira, "Smart contract: Attacks and protections," *IEEE Access,* vol. 8, pp. 24416-24427, 2020.

[11] M. Rodler, W. Li, G. O. Karame, and L. J. a. p. a. Davi, "Sereum: Protecting existing smart contracts against re-entrancy attacks," *arXiv:1812.05934,* 2018.

[12] *bZx Hack Full Disclosure (With Detailed Profit Analysis)* . Available: https://peckshield.medium.com/bzx-hack-full-disclosure-with-detailed-profit-analysis-e6b1fa9b18fc

[13] *Exploiting Uniswap: from reentrancy to actual profit*. Available: https://blog.openzeppelin.com/exploiting-uniswap-from-reentrancy-to-actual-profit/

[14] *Damn Vulnerable DeFi*. Available: https://www.damnvulnerabledefi.xyz/

[15] *MakerDAO White Paper*. Available: https://makerdao.com/en/whitepaper/#keepers

[16] *Uniswap V2 Audit Report*. Available: https://uniswap.org/audit.html#org87c8b91

[17] *Feeds price feed oracles*. Available: https://developer.makerdao.com/feeds/

[18] *Choosing a Reliable Solution for bZx's Oracle*. Available: https://bzx.network/blog/choosing-oracle