Your Exploit is Mine: Instantly Synthesizing Counterattack Smart Contract

Zhuo Zhang   Zhiqiang Lin   Marcelo Morales
Xiangyu Zhang   Kaiyuan Zhang

PURDUE UNIVERSITY   THE OHIO STATE UNIVERSITY

August 9, 2023

What Happened to Curve ?

What Happened to Curve ?

Curve, one of the most popular decentralized exchanges (DEX), was hacked for around $61.7 million on July 30th, 2023.

What Happened to `Curve` ?

`Curve`, one of the most popular decentralized exchanges (DEX), was hacked for around $61.7 million on July 30th, 2023.

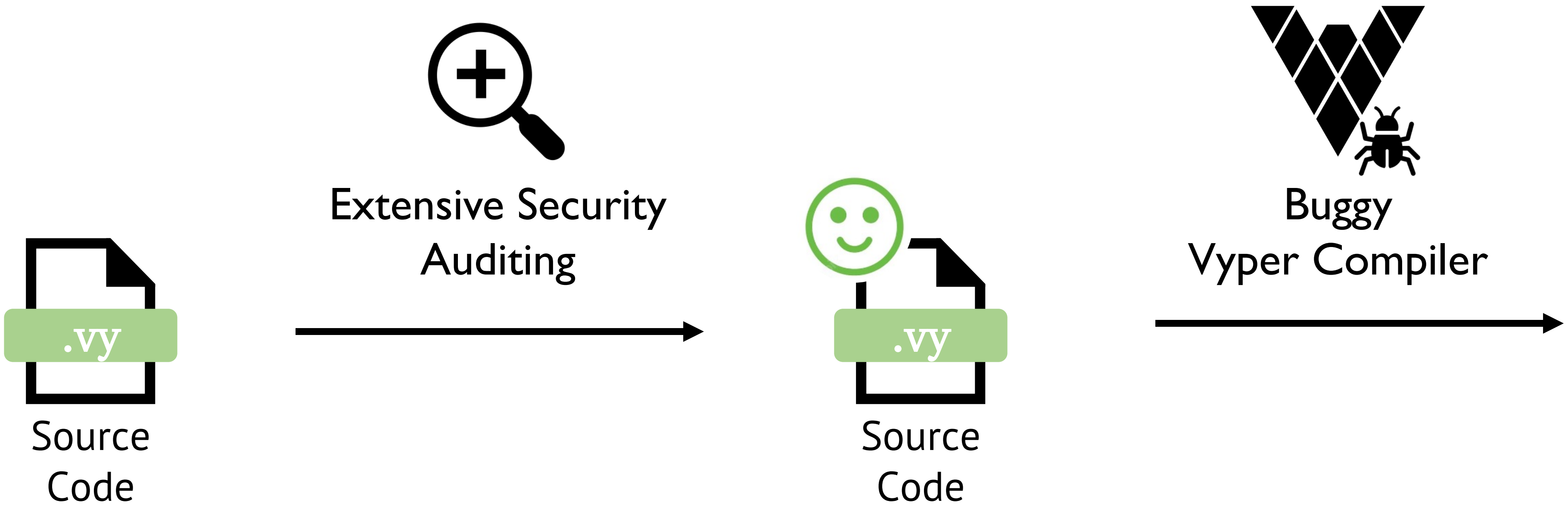
Source Code

What Happened to Curve ?

`Curve`, one of the most popular decentralized exchanges (DEX), was hacked for around $61.7 million on July 30th, 2023.

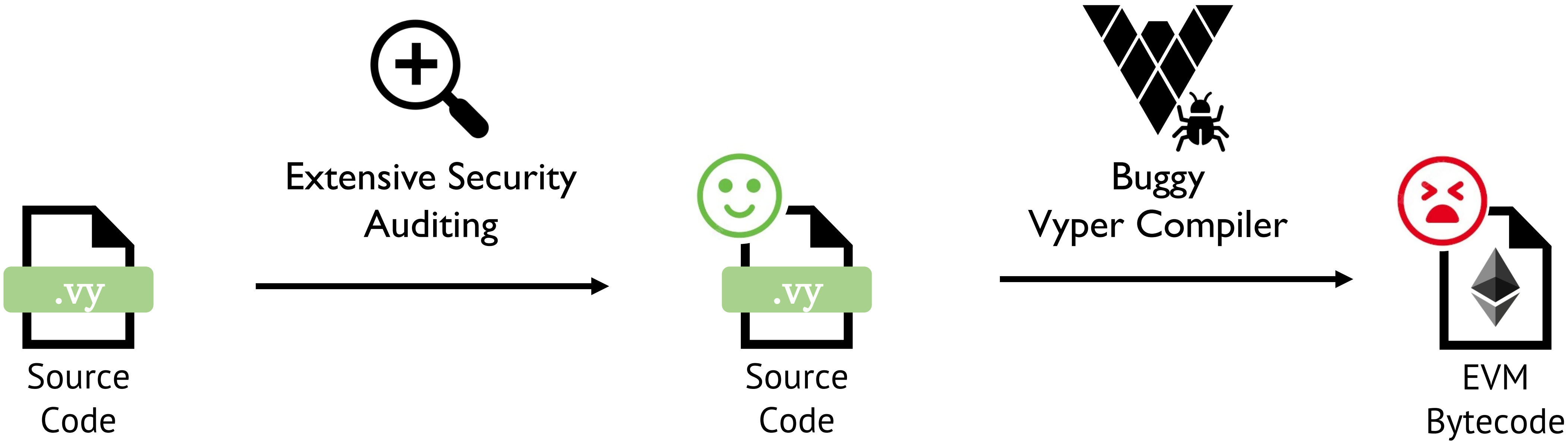
What Happened to Curve ?

Curve, one of the most popular decentralized exchanges (DEX), was hacked for around $61.7 million on July 30th, 2023.



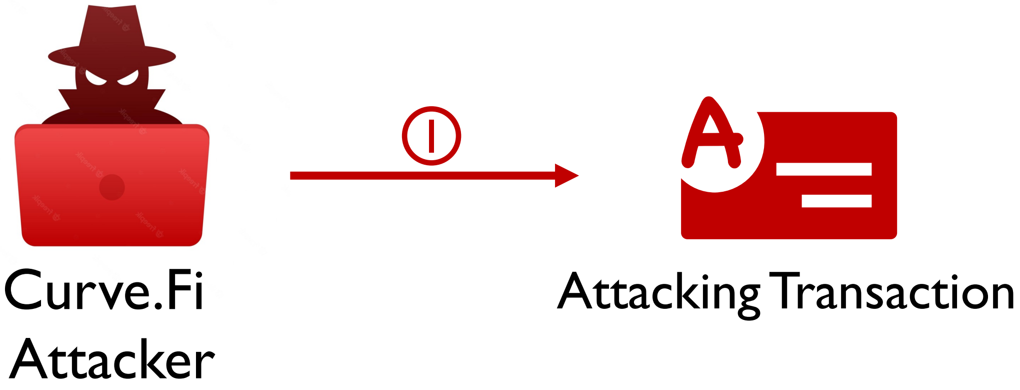
What Happened to Curve ?

Curve, one of the most popular decentralized exchanges (DEX), was hacked for around $61.7 million on July 30th, 2023.

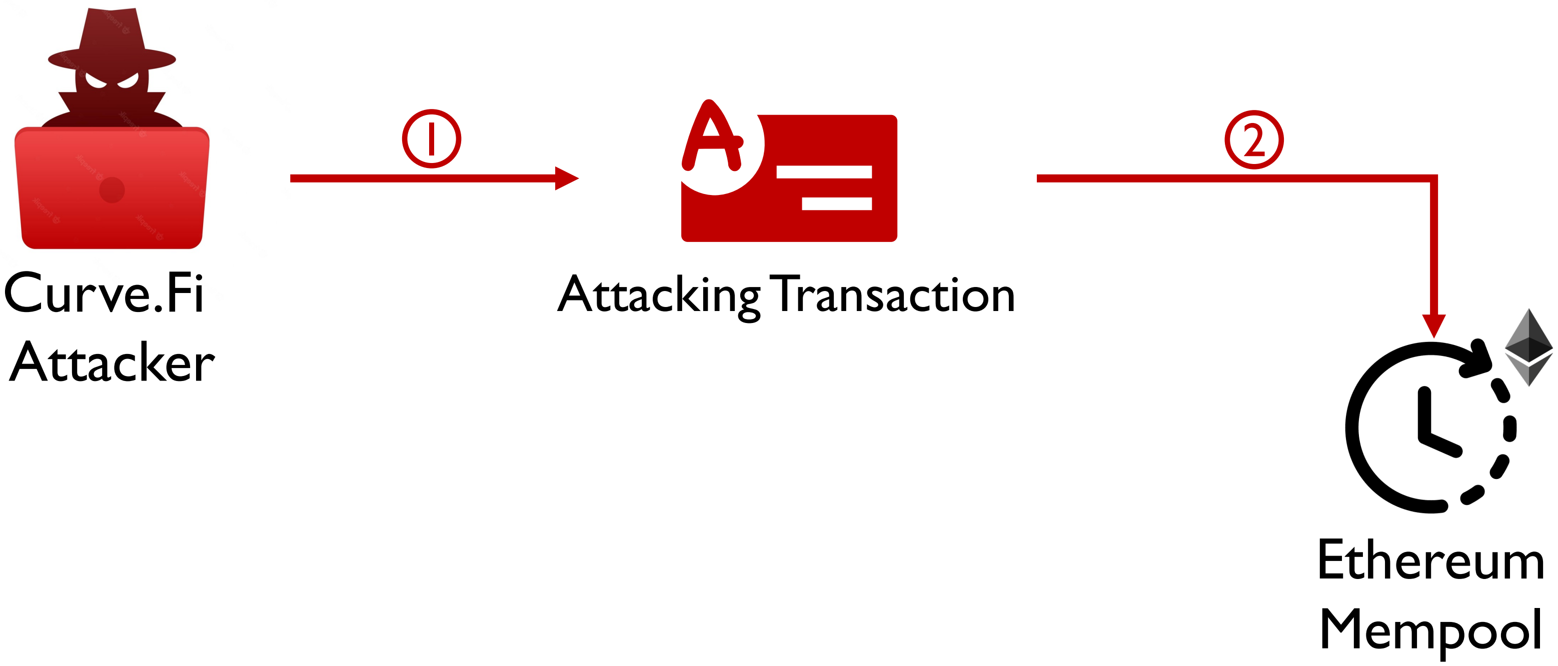


What Happened to Curve ?

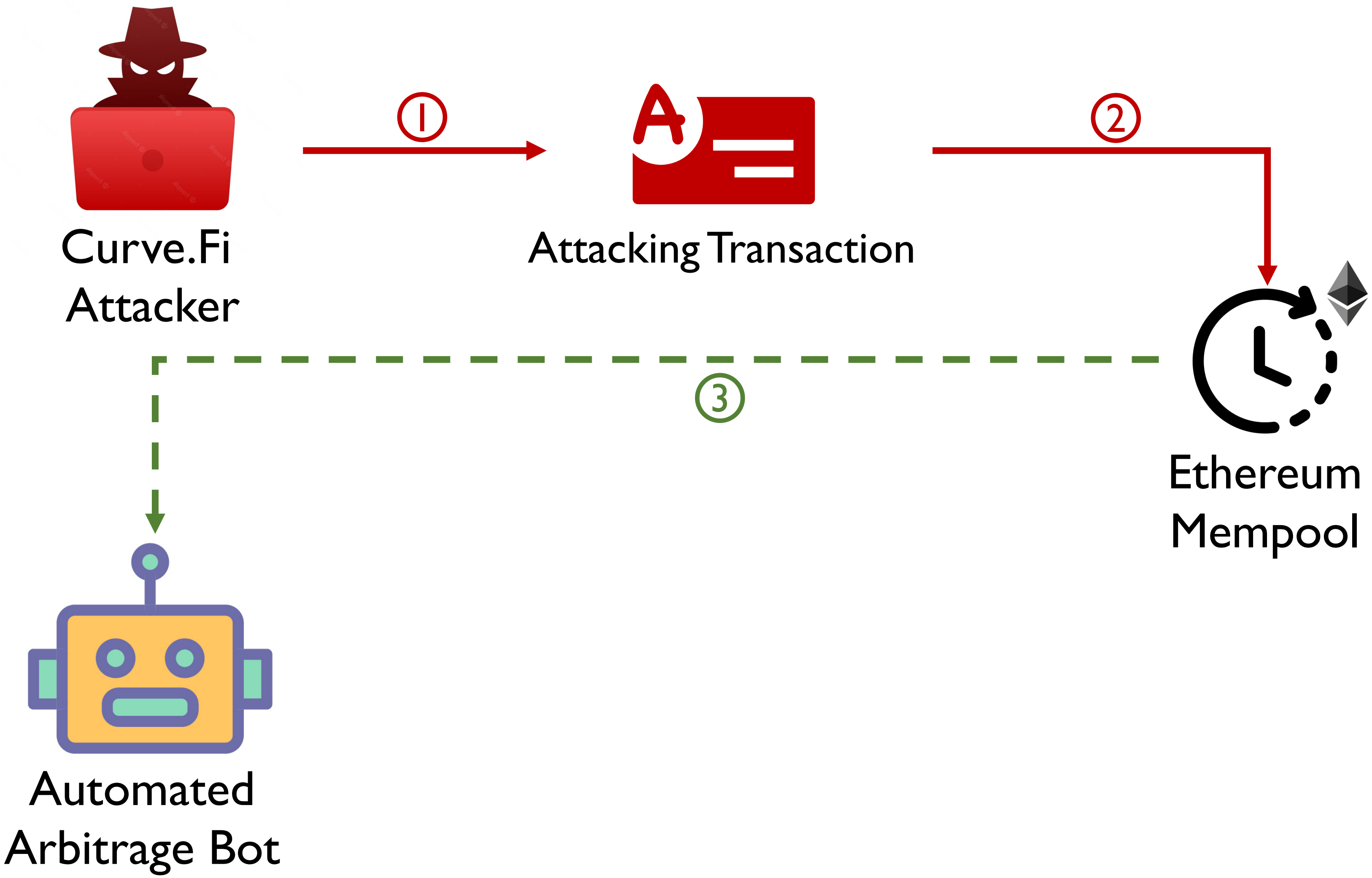
What Happened to Curve ?



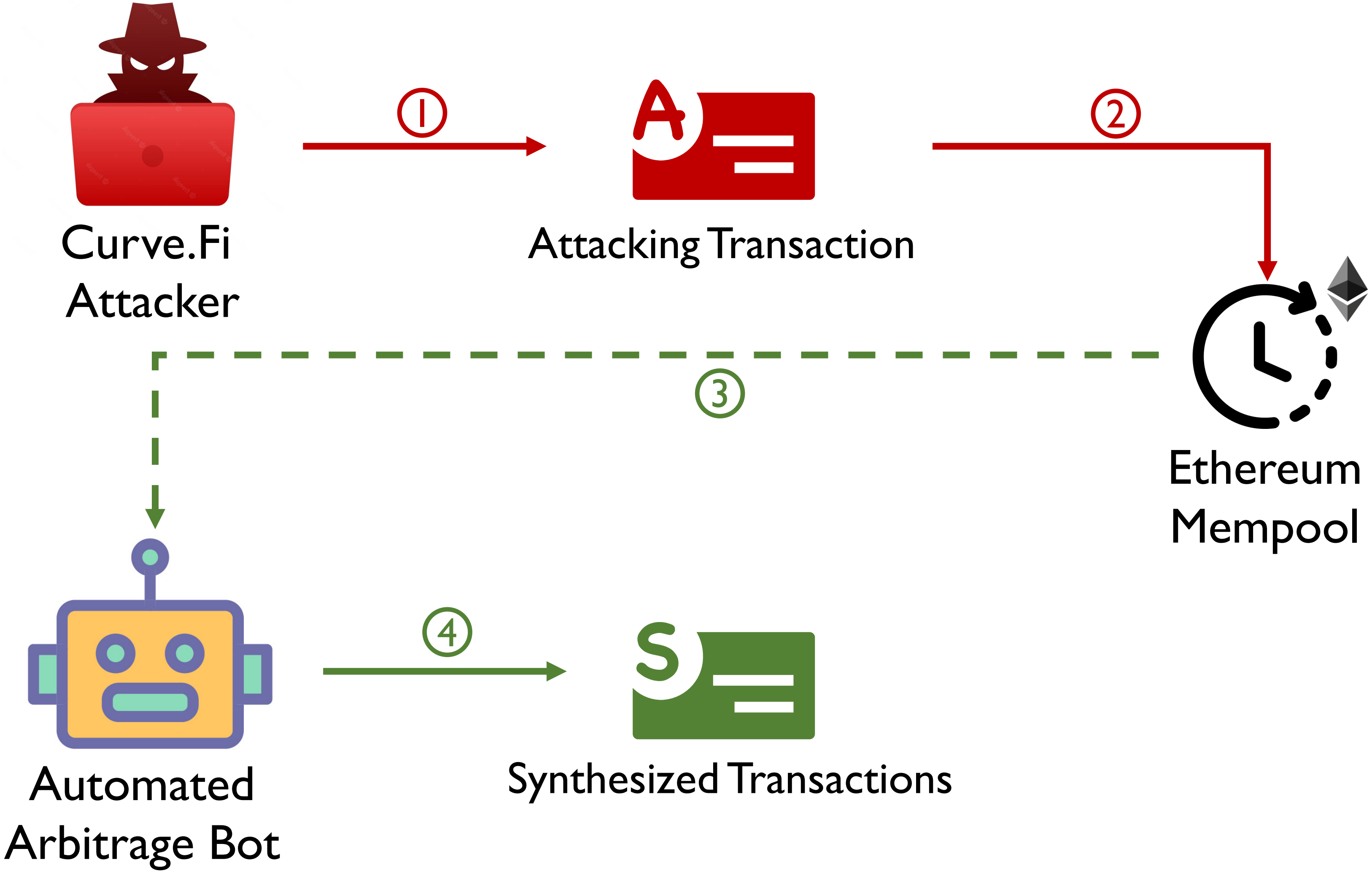
What Happened to Curve ?



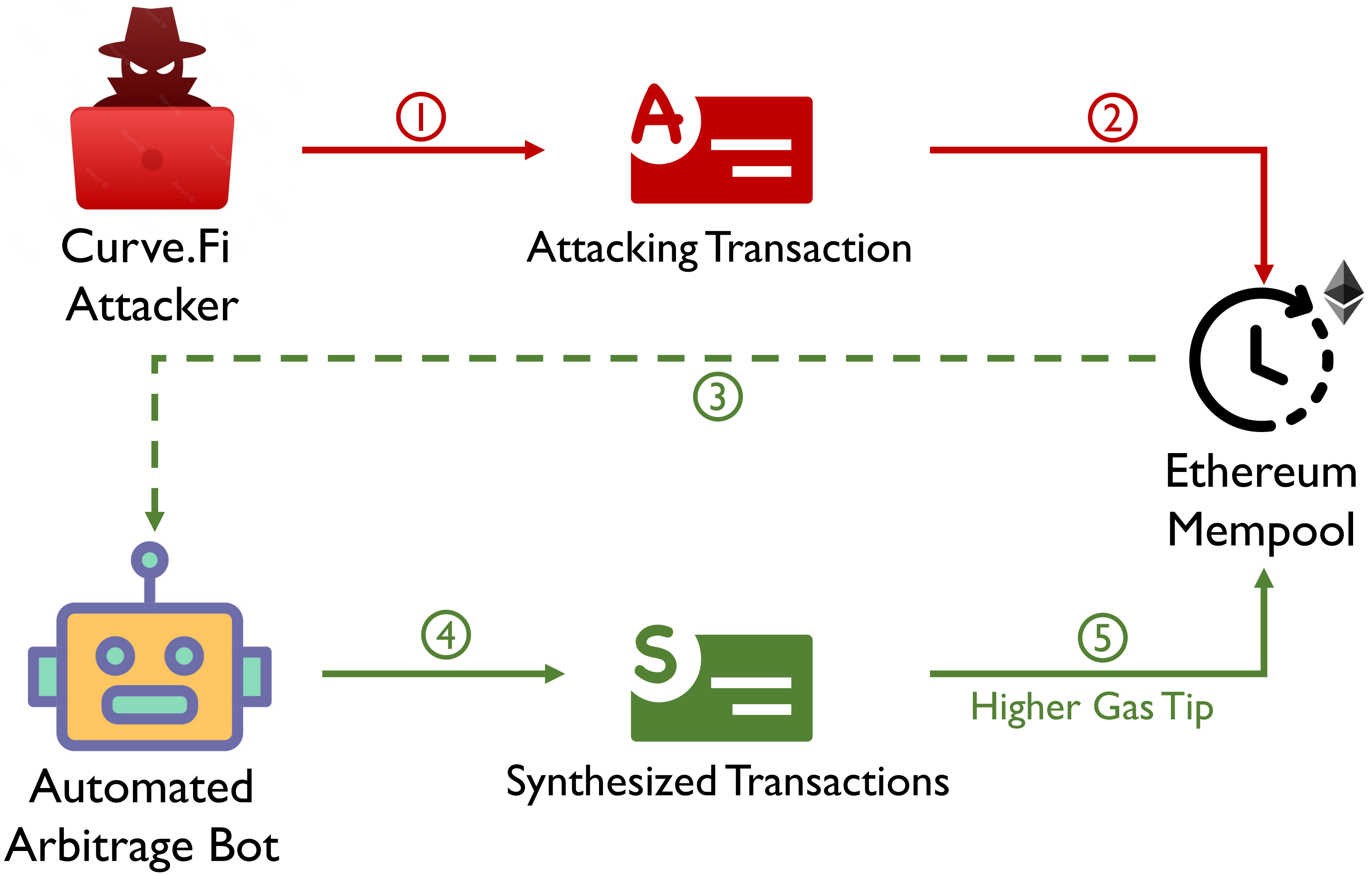
What Happened to Curve ?



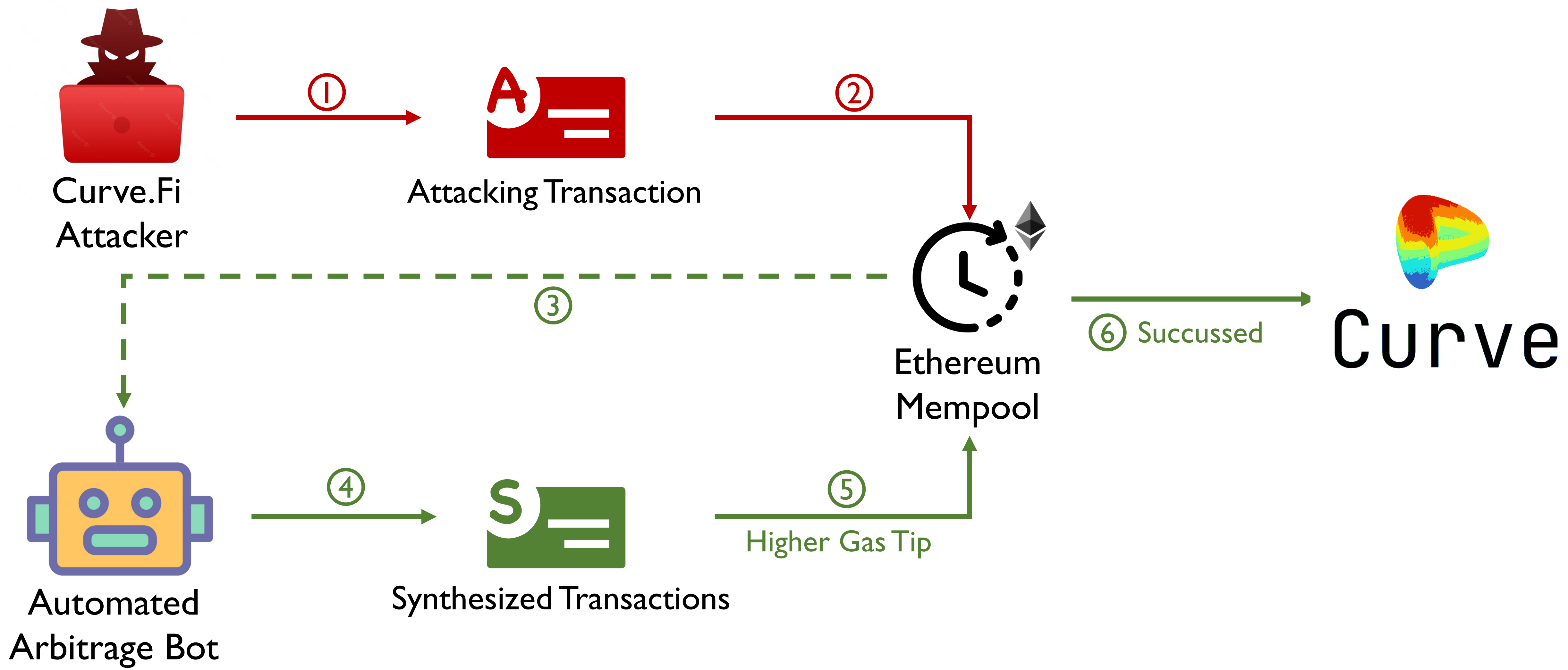
What Happened to Curve ?



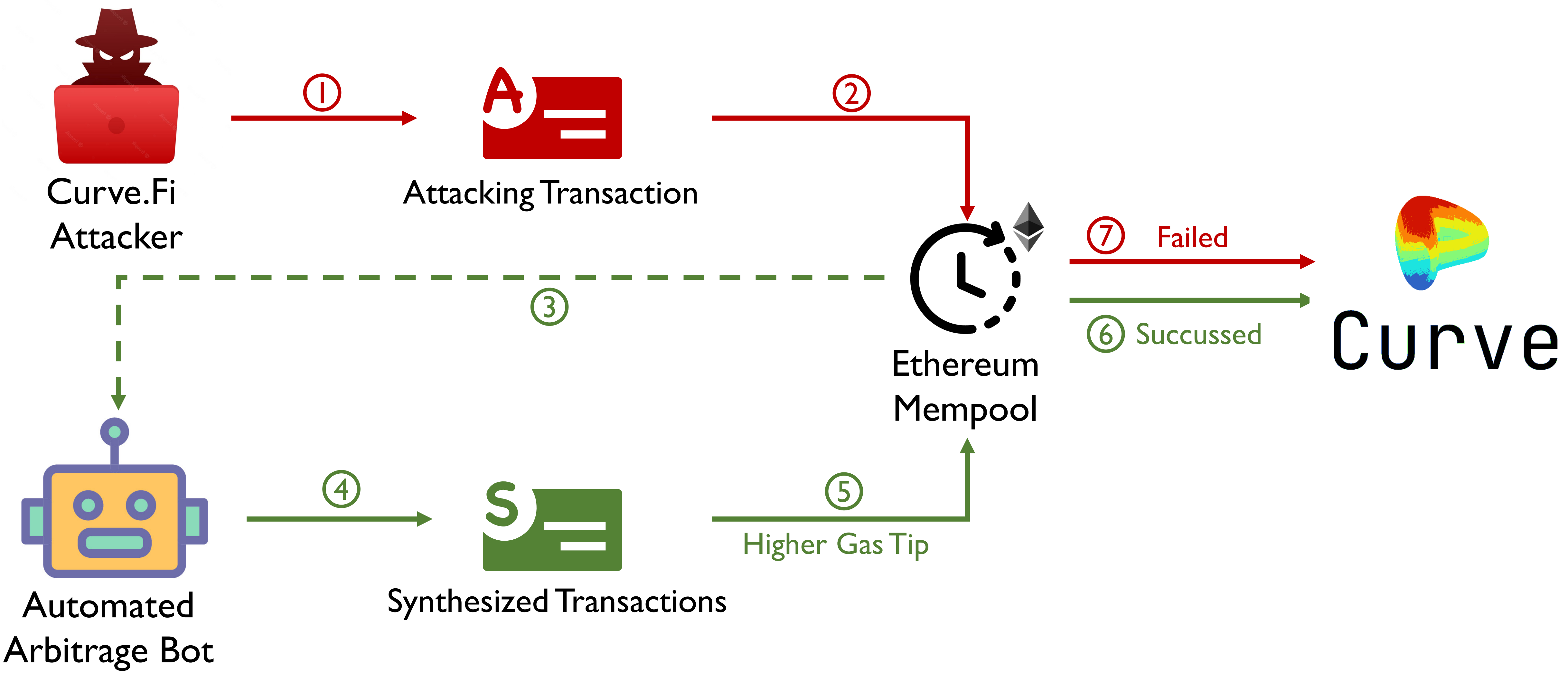
What Happened to Curve ?



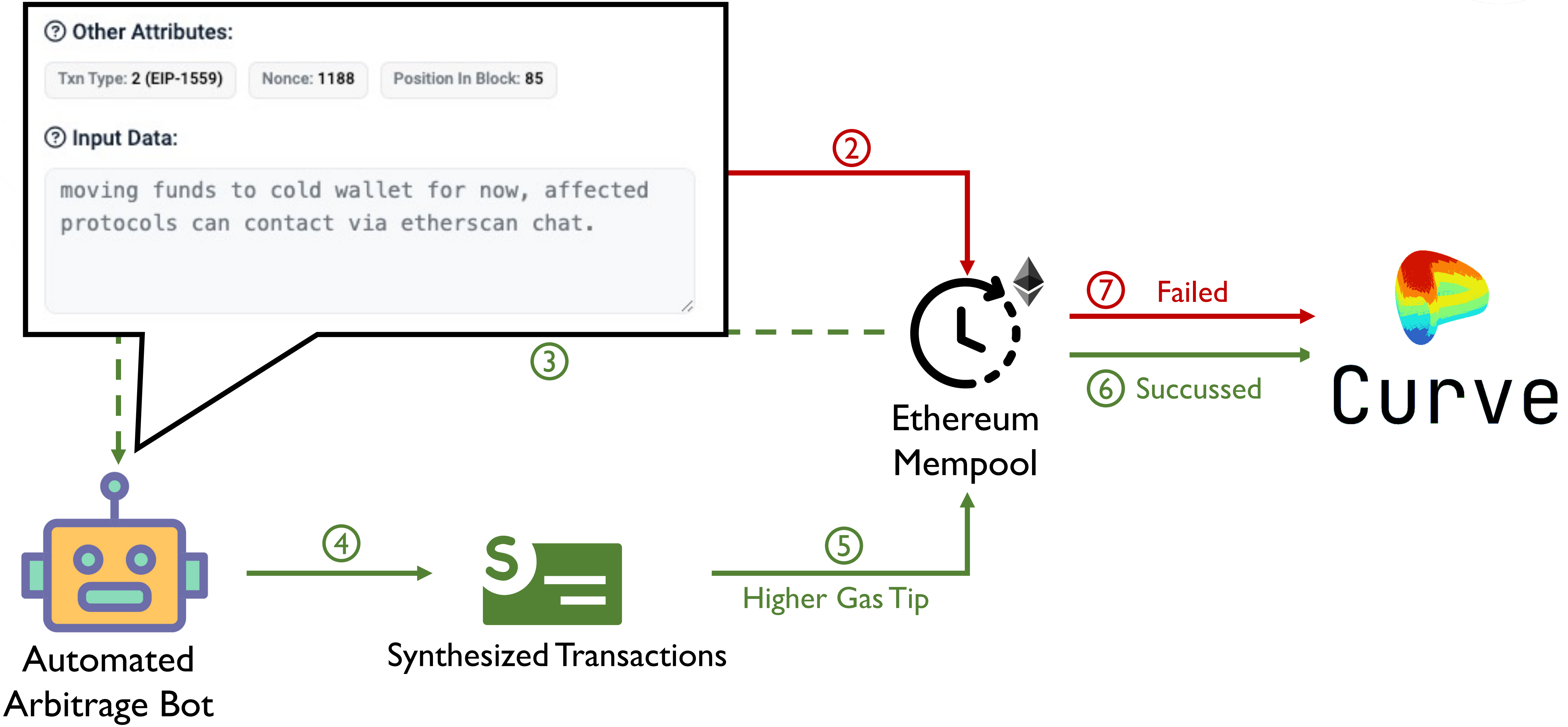
What Happened to Curve ?



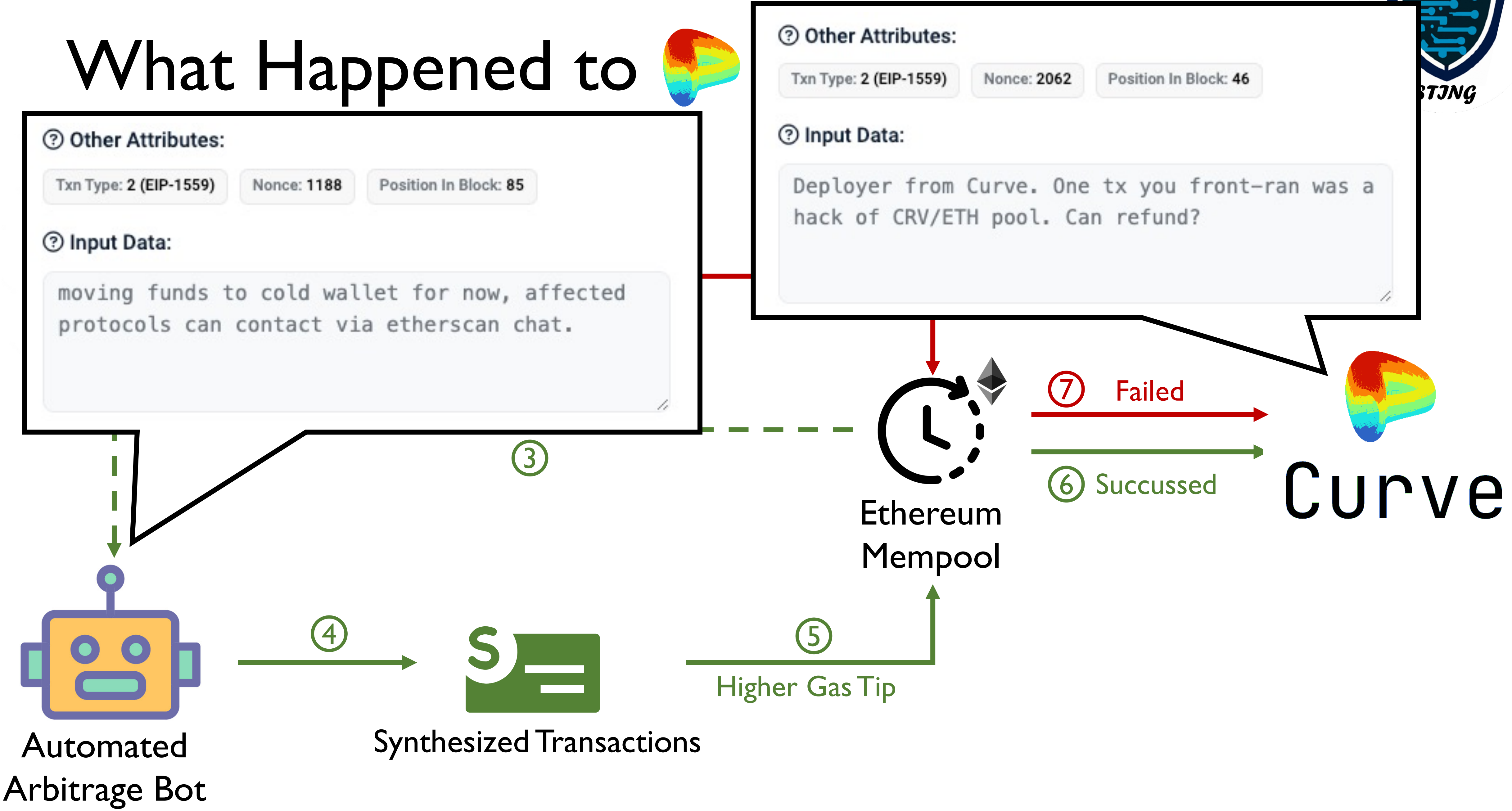
What Happened to Curve ?



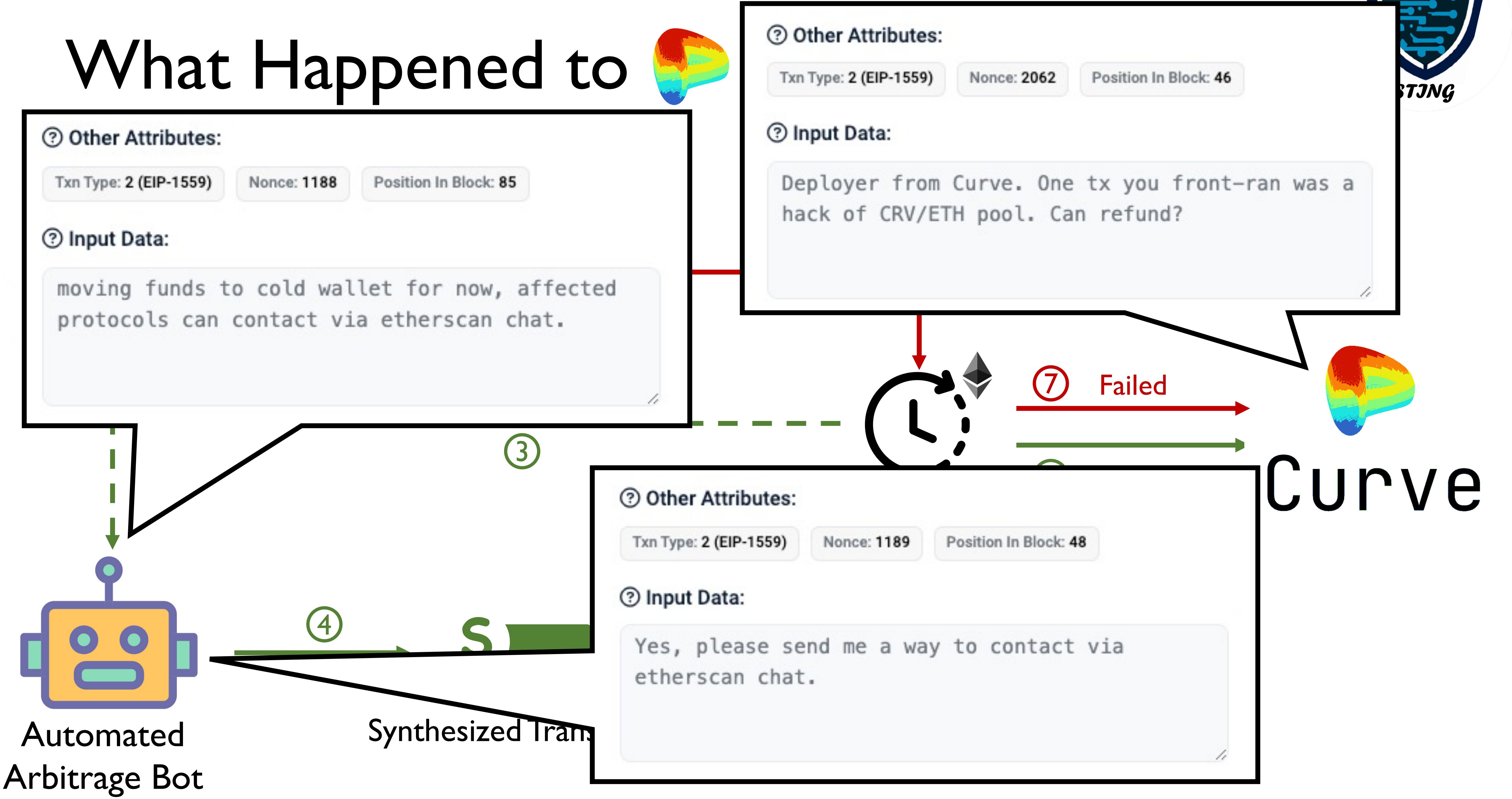
What Happened to Curve ?



What Happened to



What Happened to



Takeaways from Curve

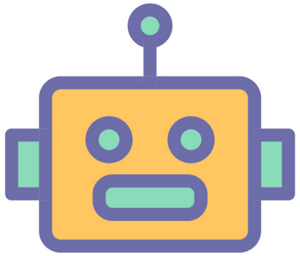
Takeaways from Curve

- Many vulnerabilities may remain hidden despite a large amount of auditing efforts having been put forth.
- Frontrunning attacking transactions provides another opportunity to protect user funds.

Buggy
Vyper Compiler

Takeaways from Curve

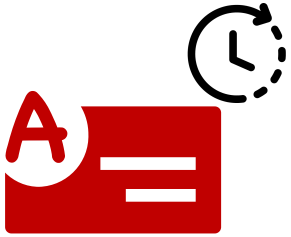
- Many vulnerabilities may remain hidden despite a large amount of auditing efforts having been put forth.
- Frontrunning attacking transactions provides another opportunity to protect user funds.



Automated Arbitrage Bot

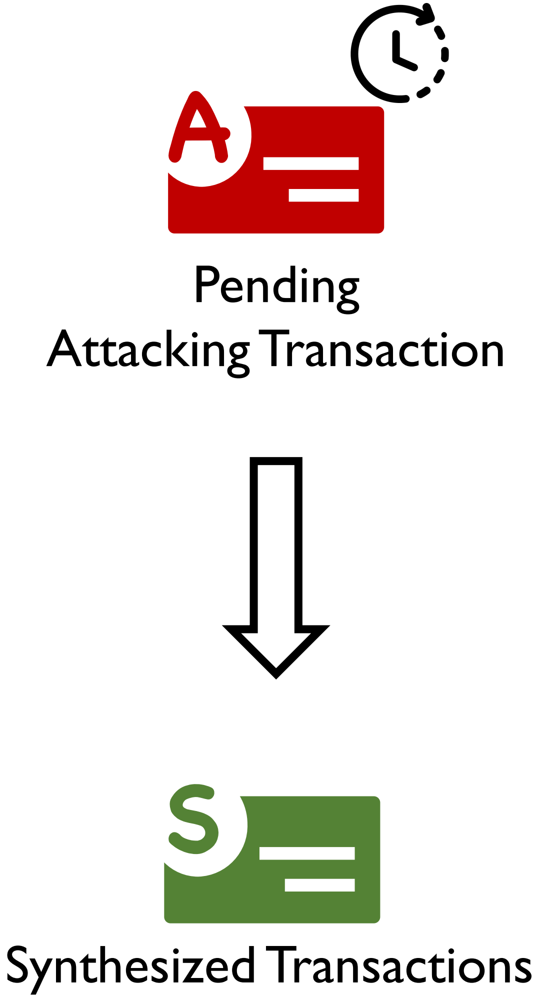
The Goal and the Timeline

The Goal and the Timeline

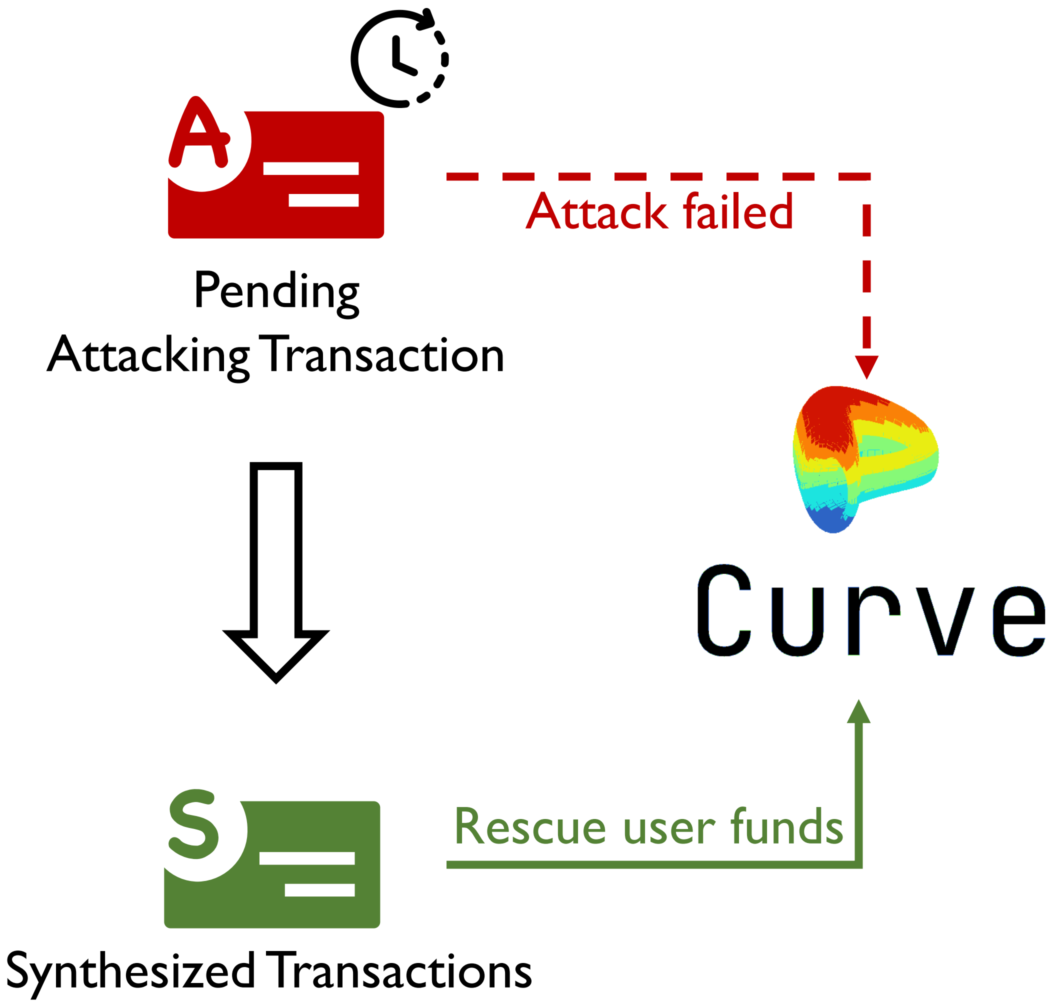


Pending
Attacking Transaction

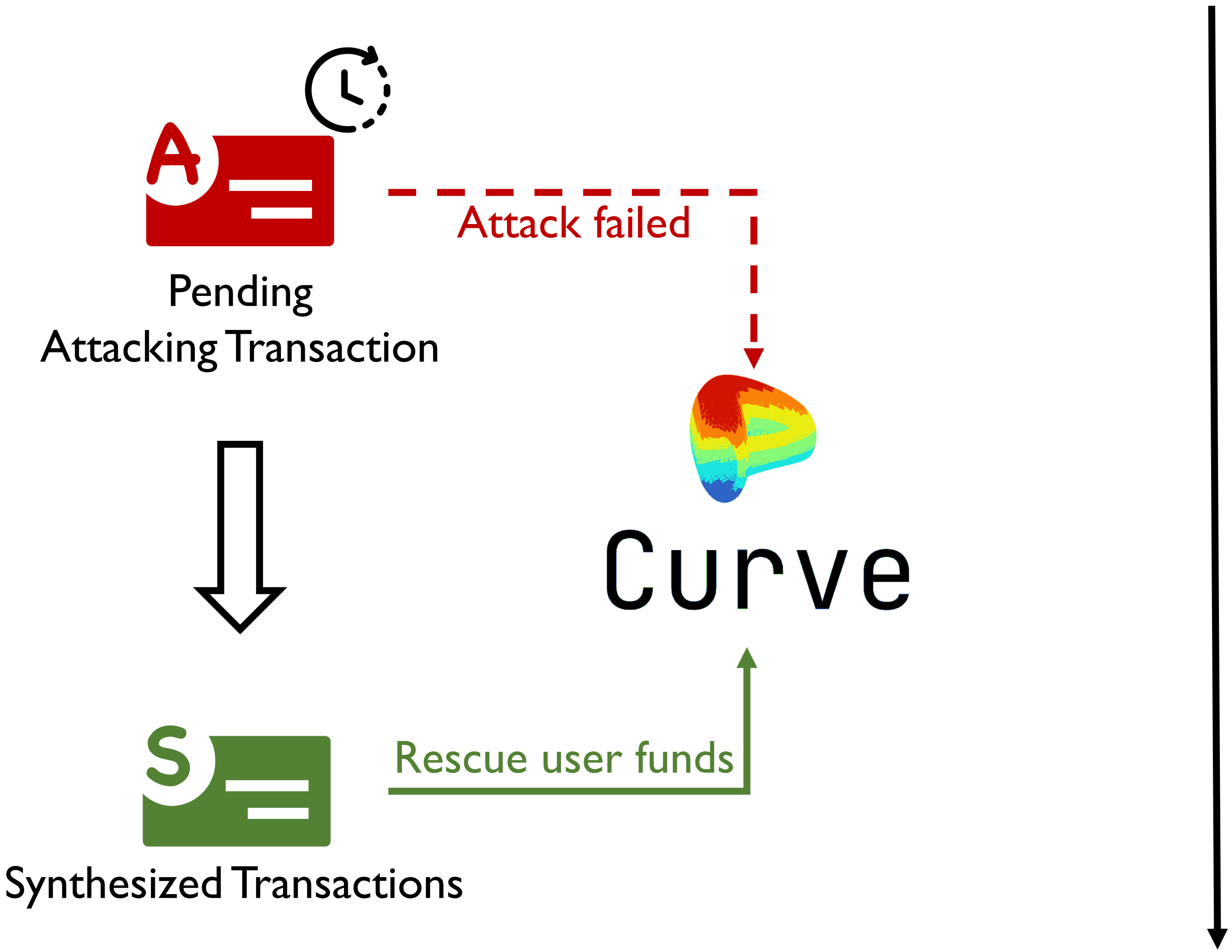
The Goal and the Timeline



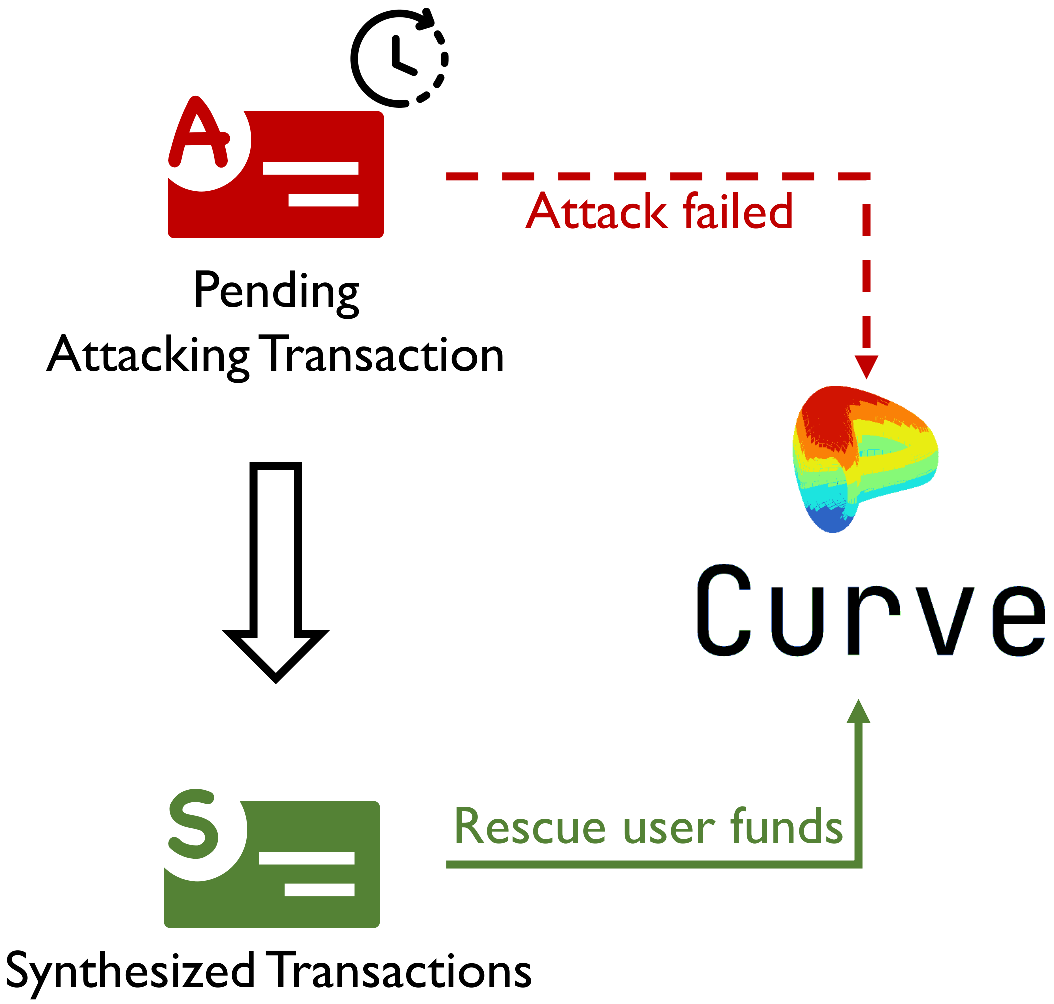
The Goal and the Timeline



The Goal and the Timeline

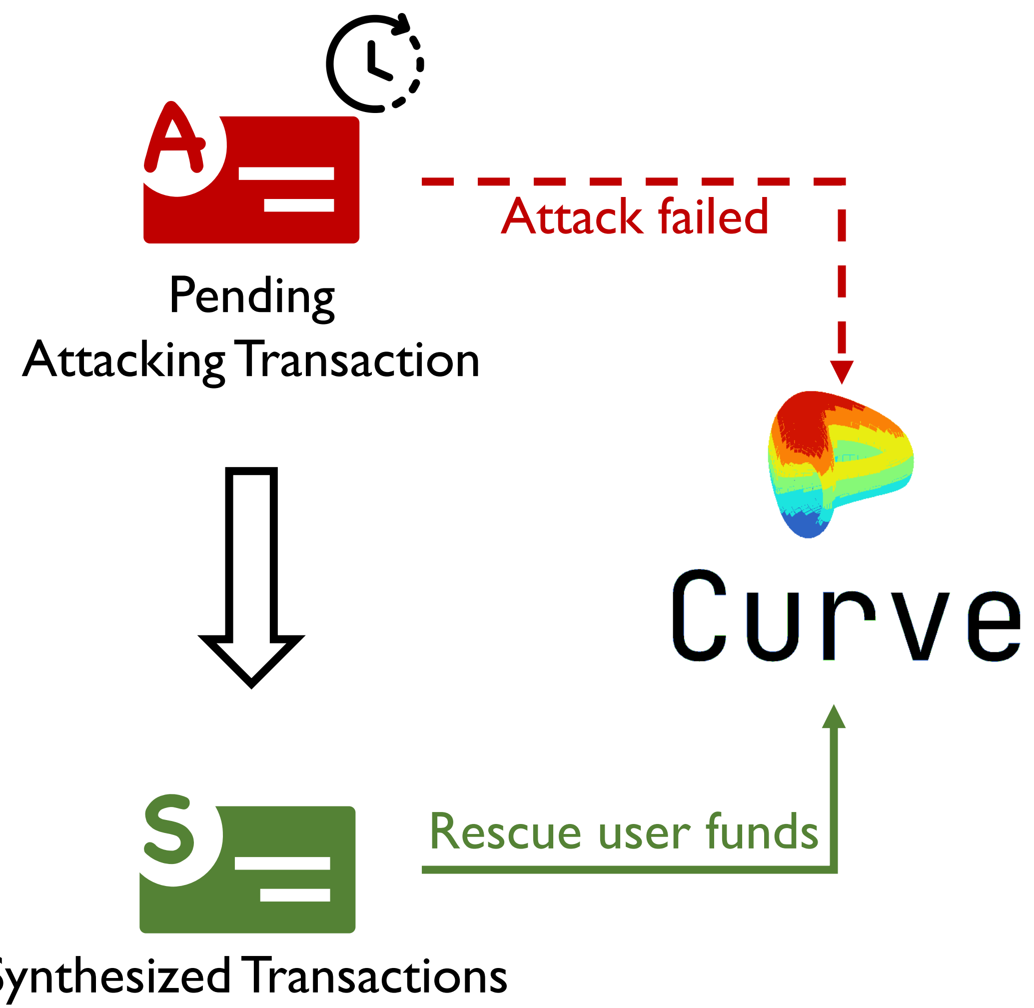


The Goal and the Timeline



2021  *OfficerCia*, a Twitter user proposed a similar idea.

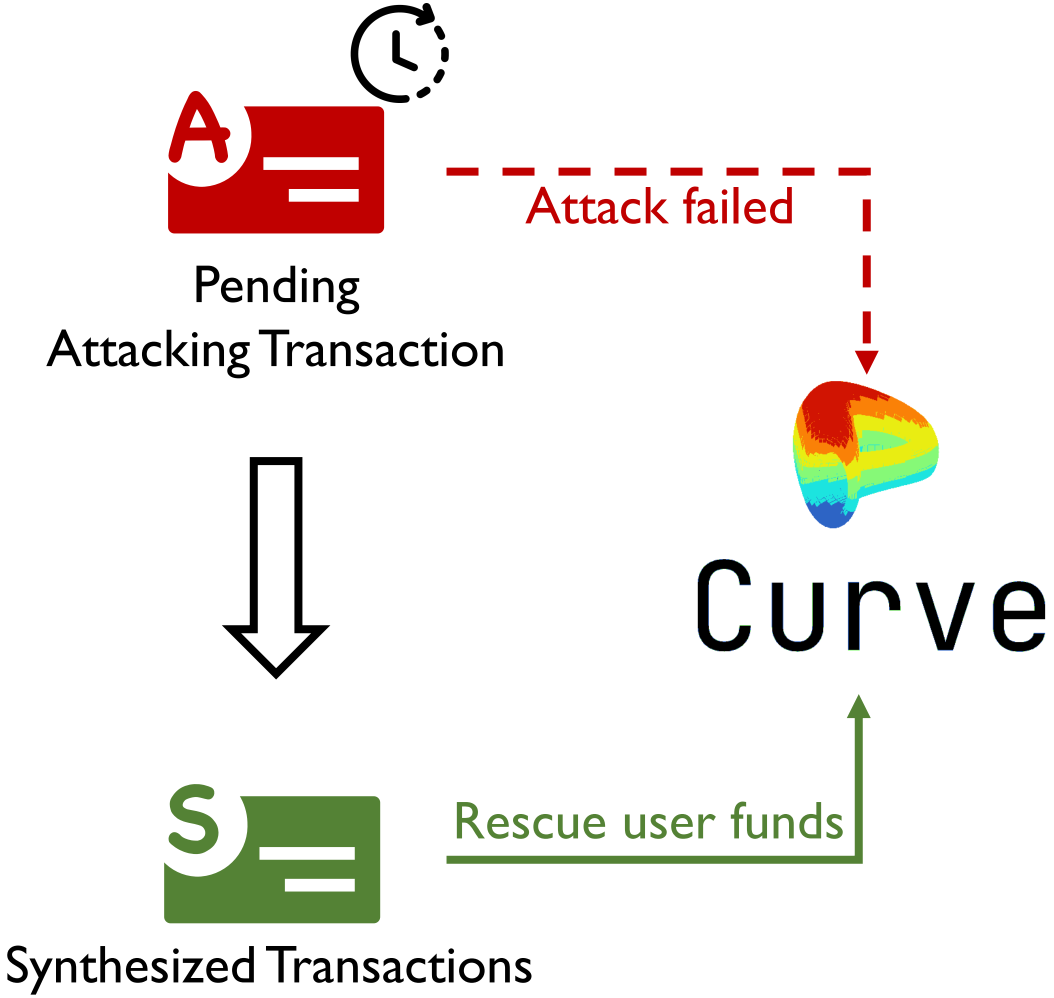
The Goal and the Timeline



2021 · *OfficerCia*, a Twitter user proposed a similar idea.

2022 · *BlockSec*, a DeFi security company, successfully prevented a real-world attack, rescuing around $3.8 million.

The Goal and the Timeline

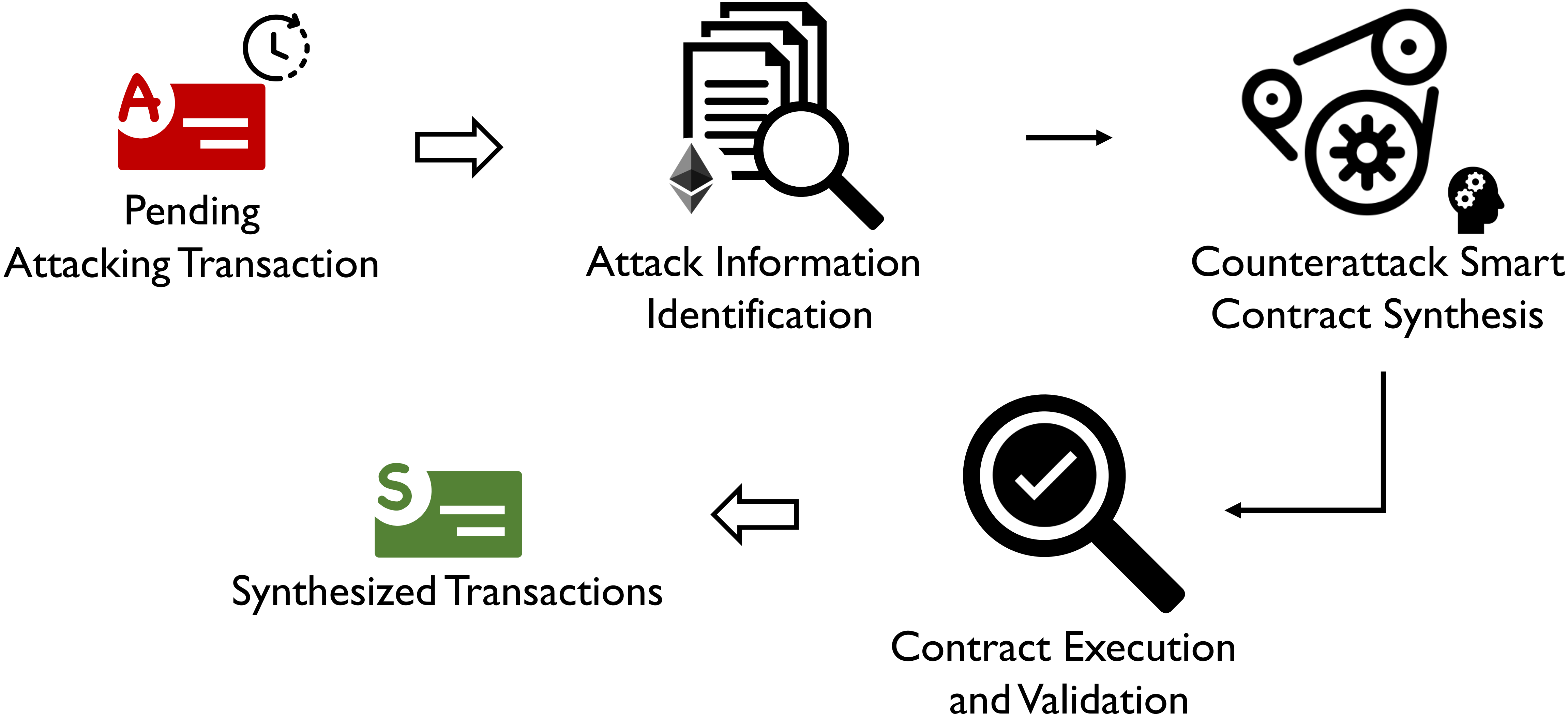


2021 *OfficerCia*, a Twitter user proposed a similar idea.

2022 *BlockSec*, a DeFi security company, successfully prevented a real-world attack, rescuing around $3.8 million.

2023 Many well-known DeFi security companies have started to put their efforts into this arena: *BlockSec*, *FuzzLand*, *Skylock*, *D23E.ch*, *Spotter*, and more.

Our Solution: STING



Running Example[1]

Vulnerable Contract

```
contract Victim {
  address operator;

  function setOperator(address _operator) {
    operator = _operator;
  }

  function emergencyExit(address to) {
    require(operator == msg.sender);
    to.transfer(address(this).balance);
  }
}
```

[1] For illustrative purposes, the example is a crafted one, combining *DAOMaker* and *TempleDao* exploits.

Running Example[1]

Vulnerable Contract

```
contract Victim {
  address operator;

  function setOperator(address _operator) {
    operator = _operator;
  }

  function emergencyExit(address to) {
    require(operator == msg.sender);
    to.transfer(address(this).balance);
  }
}
```

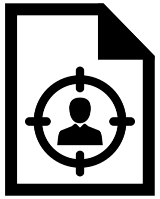
[1] For illustrative purposes, the example is a crafted one, combining *DAOMaker* and *TempleDao* exploits.

Running Example[1]

Vulnerable Contract

```
contract Victim {
  address operator;

  function setOperator(address _operator) {
    operator = _operator;
  }

  function emergencyExit(address to) {
    require(operator == msg.sender);
    to.transfer(address(this).balance);
  }
}
```

Attacker



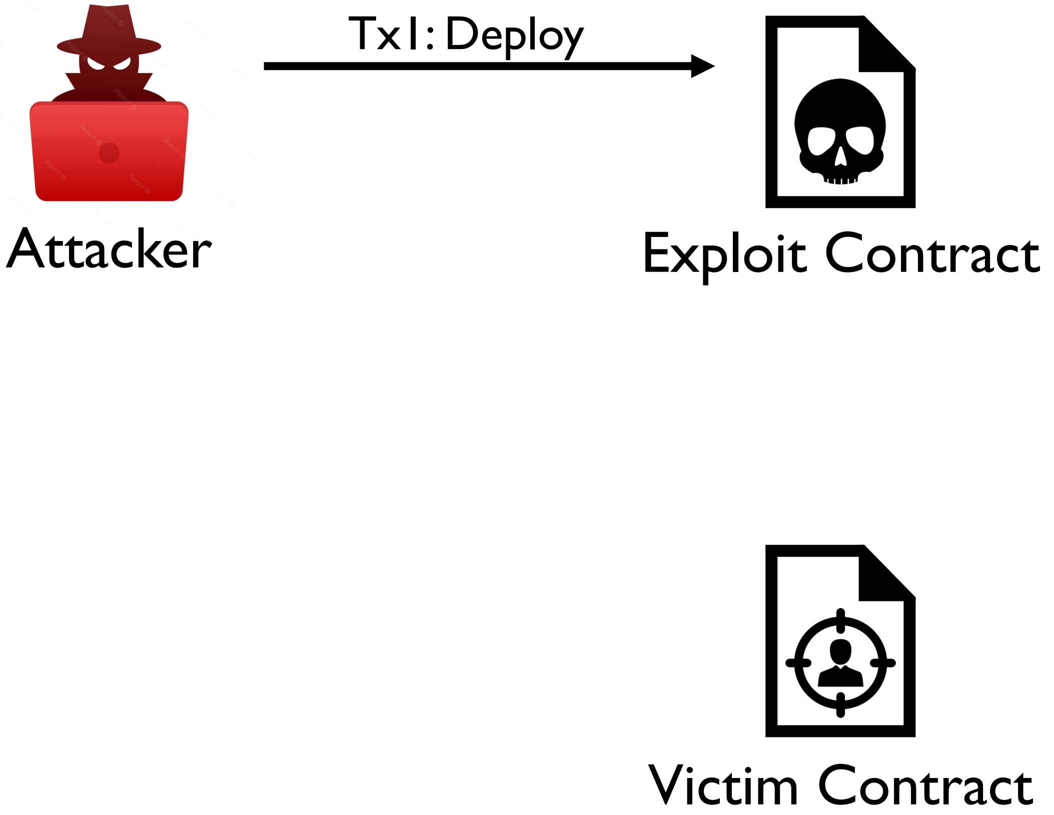
Victim Contract

[1] For illustrative purposes, the example is a crafted one, combining *DAOMaker* and *TempleDao* exploits.

Running Example[1]

Vulnerable Contract

```
contract Victim {
  address operator;

  function setOperator(address _operator) {
    operator = _operator;
  }

  function emergencyExit(address to) {
    require(operator == msg.sender);
    to.transfer(address(this).balance);
  }
}
```

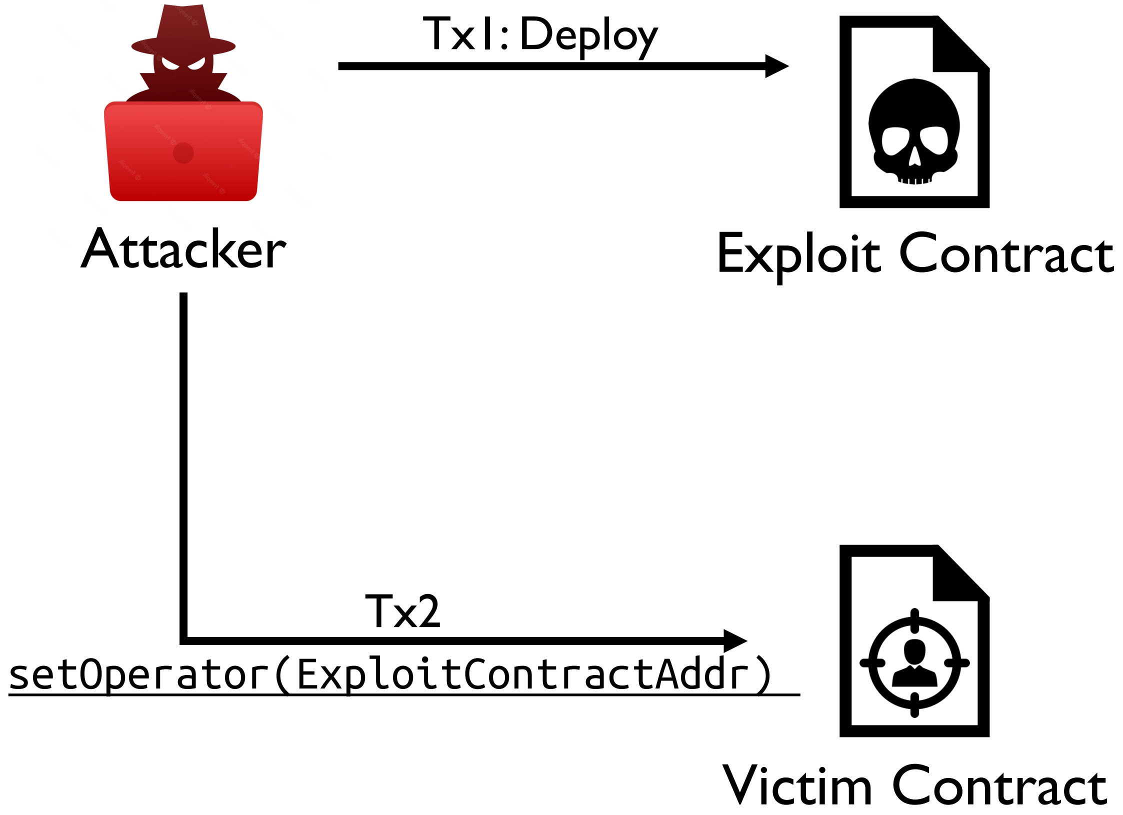


[1] For illustrative purposes, the example is a crafted one, combining DAOMaker and TempleDao exploits.

Running Example[1]

Vulnerable Contract

```
contract Victim {
  address operator;

  function setOperator(address _operator) {
    operator = _operator;
  }

  function emergencyExit(address to) {
    require(operator == msg.sender);
    to.transfer(address(this).balance);
  }
}
```

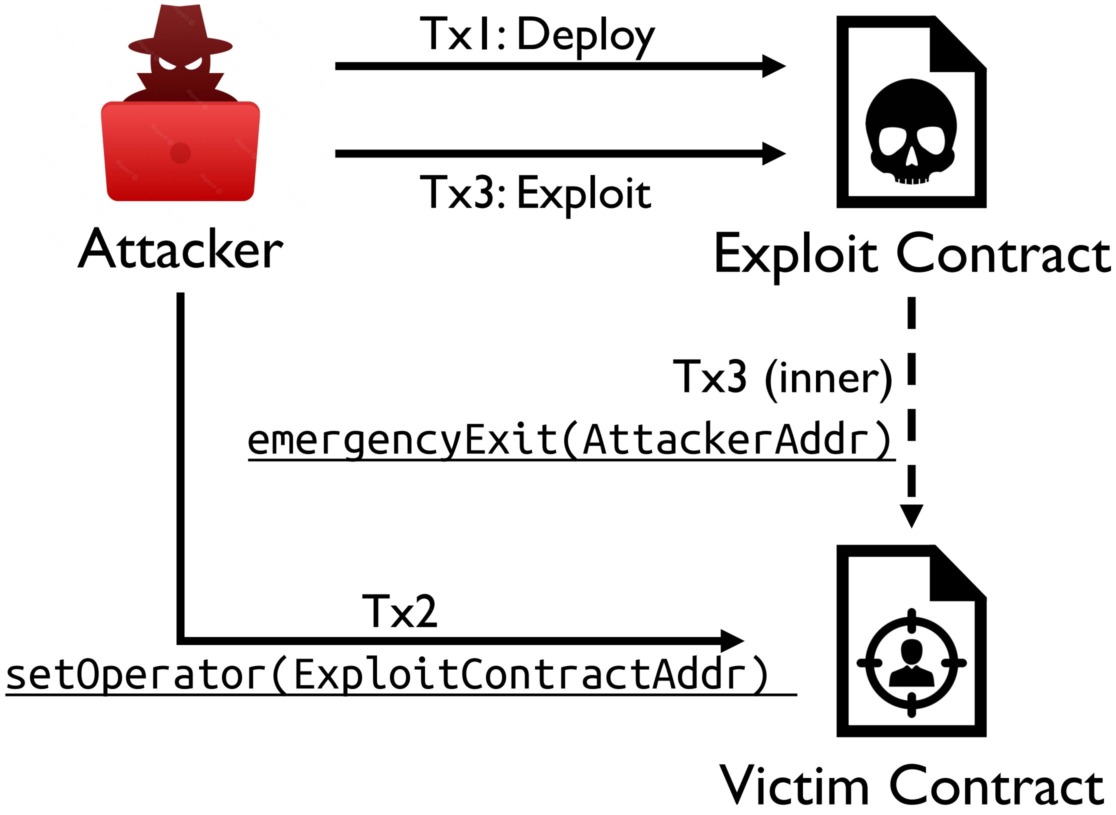


[1] For illustrative purposes, the example is a crafted one, combining *DAOMaker* and *TempleDao* exploits.

Running Example[1]

Vulnerable Contract

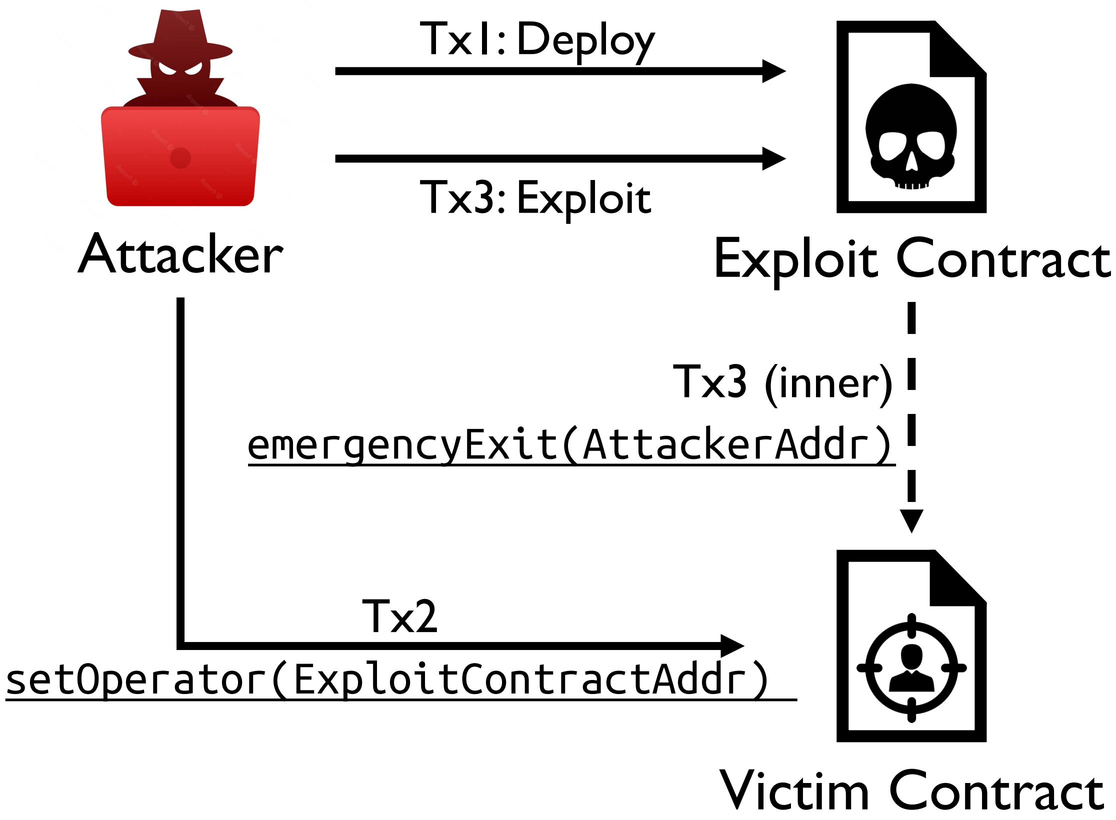
```
contract Victim {
  address operator;

  function setOperator(address _operator) {
    operator = _operator;
  }

  function emergencyExit(address to) {
    require(operator == msg.sender);
    to.transfer(address(this).balance);
  }
}
```



[1] For illustrative purposes, the example is a crafted one, combining DAOMaker and TempleDao exploits.

Attack Information Identification

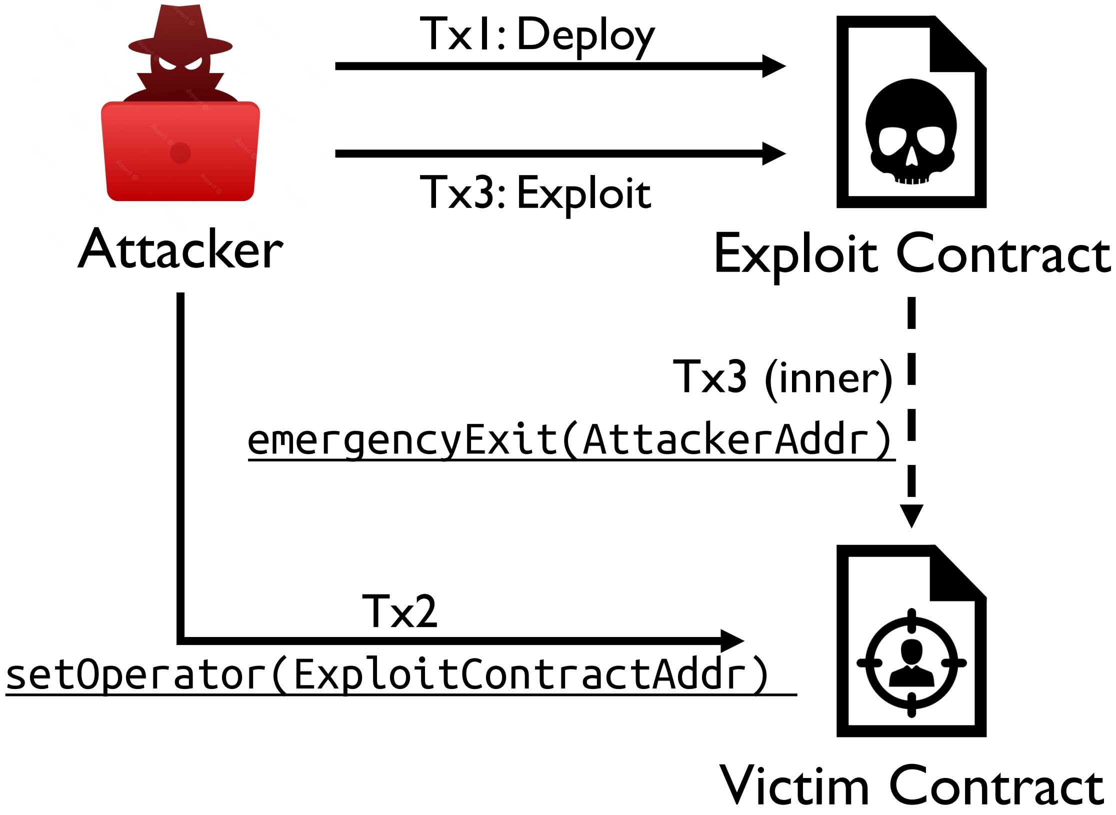
Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.



Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

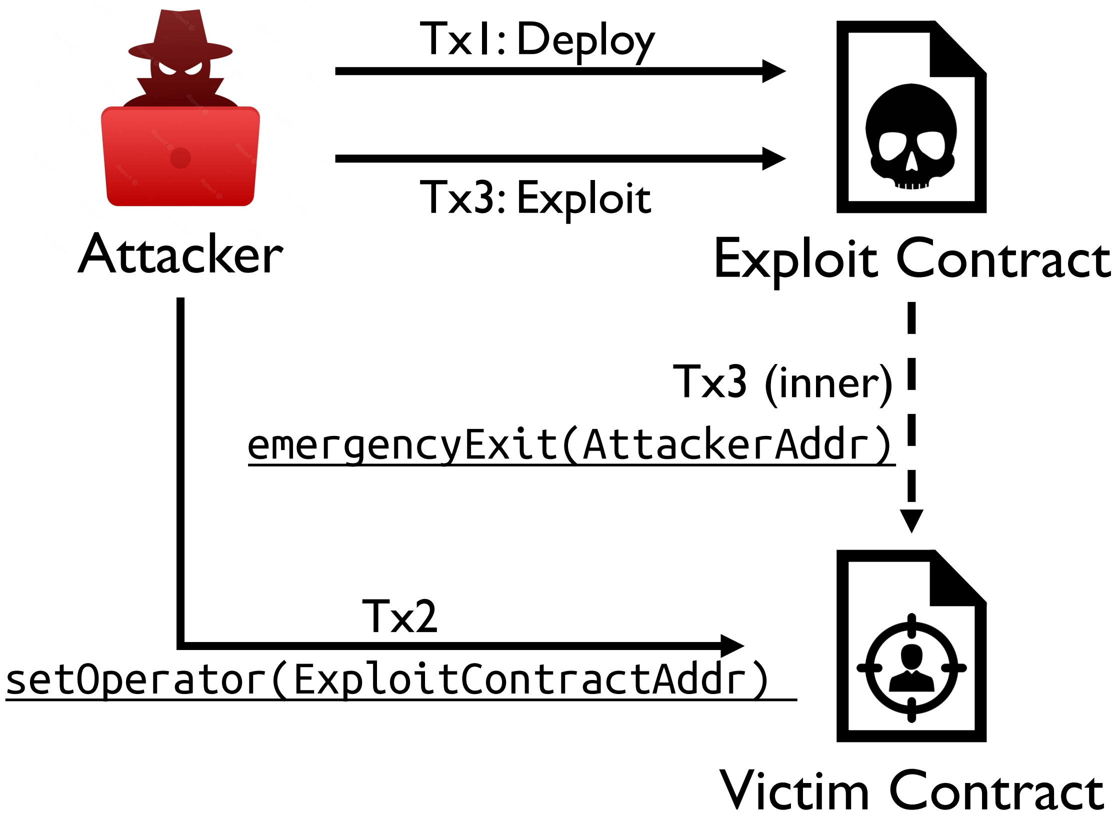


Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

- The attacking transaction is detected based on the profit.

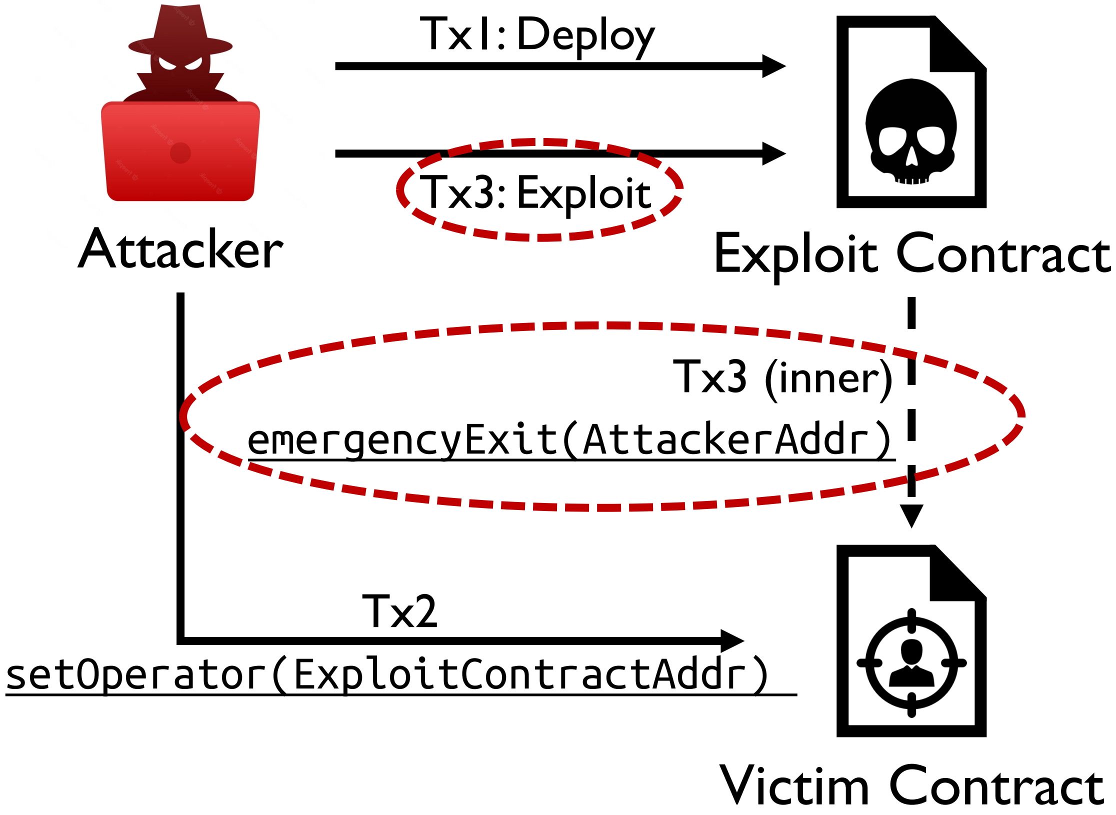


Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

- The attacking transaction is detected based on the profit.

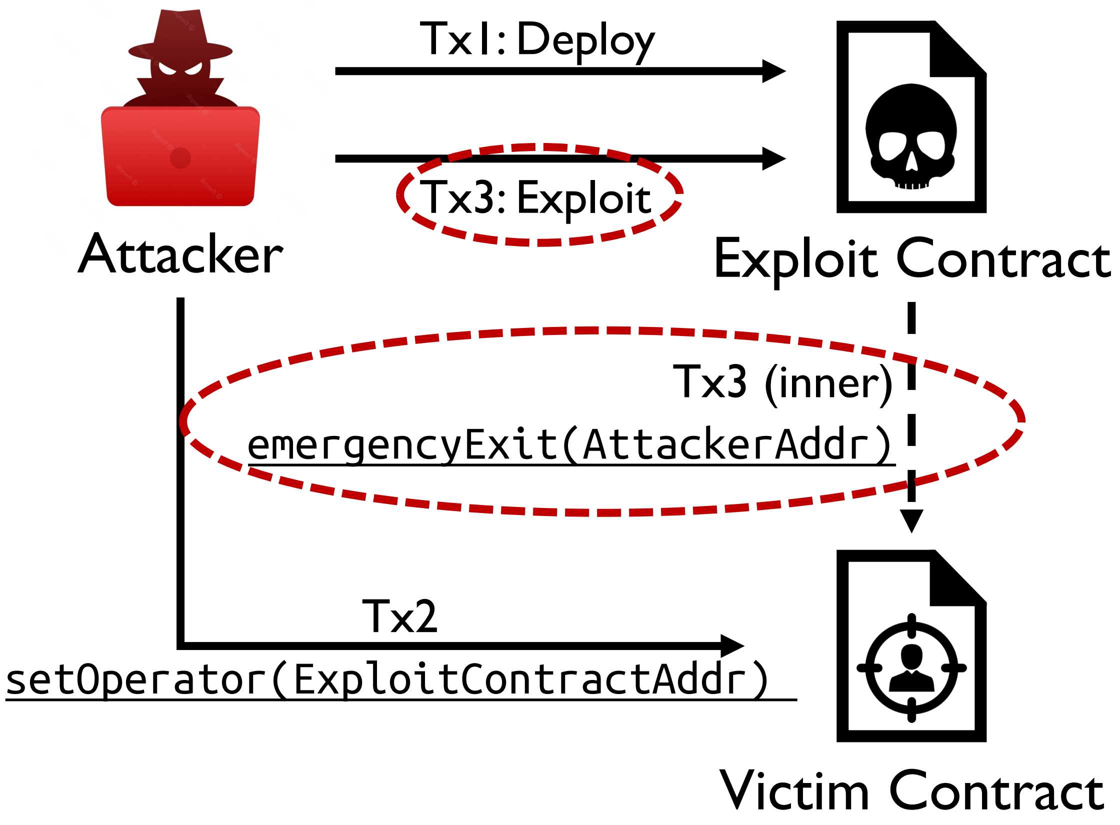


Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

- The attacking transaction is detected based on the profit.
- Accounts are pinpointed based on historical behaviors.

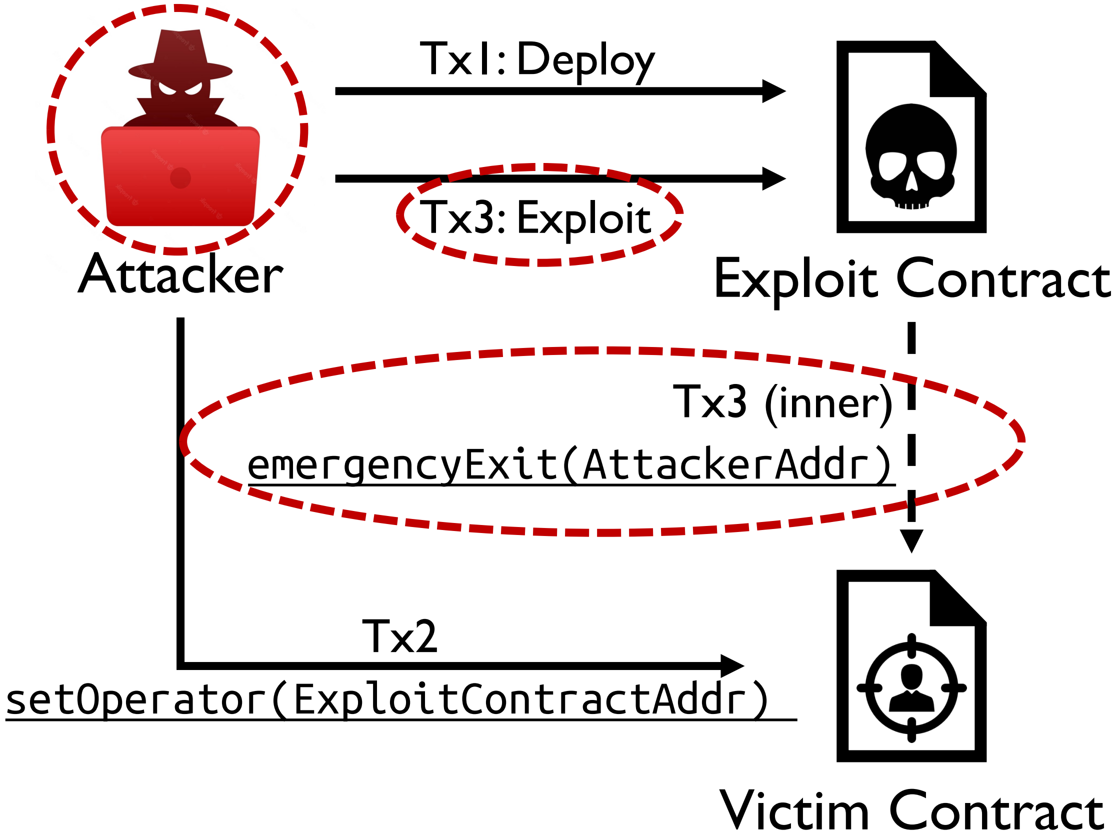


Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

- The attacking transaction is detected based on the profit.
- Accounts are pinpointed based on historical behaviors.



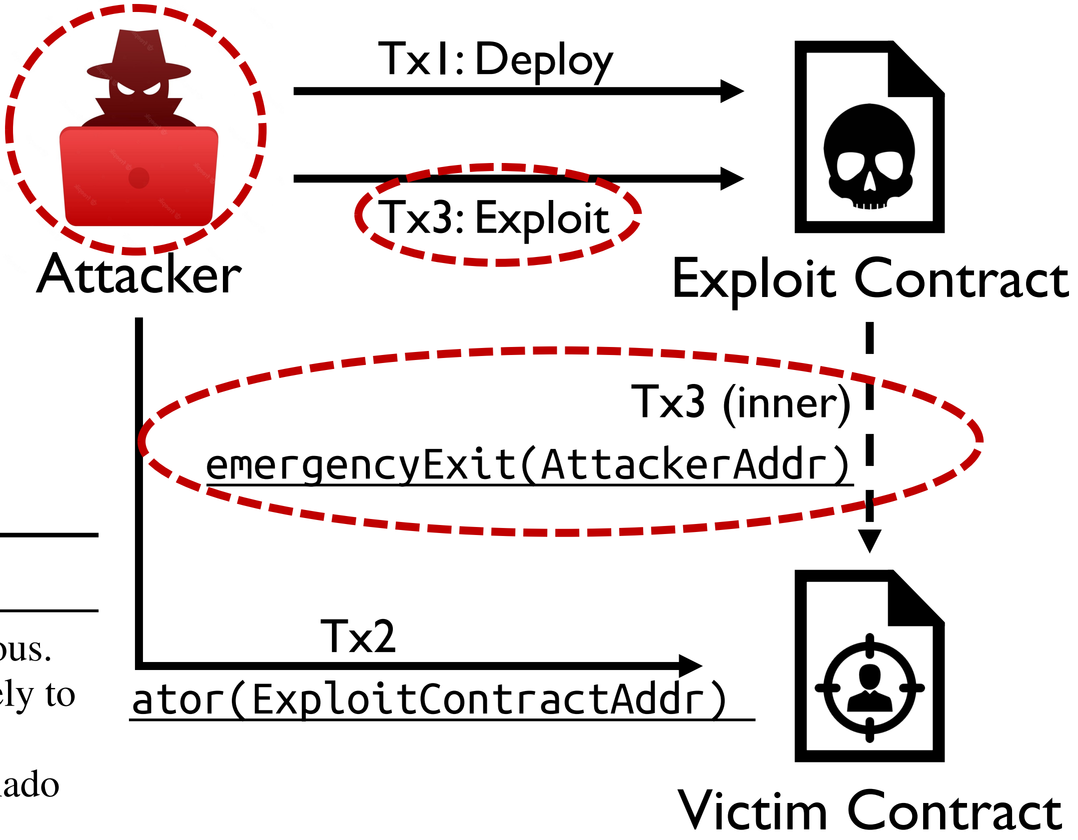
Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

- The attacking transaction is detected based on the profit.

Name	Description
Lifespan	Contracts deployed shortly before an attack are likely malicious.
Balance	Contracts whose initial assets exceed the attack profit are likely to be victims.
Fund Source	Contracts and Wallets funded from mixing servers (e.g., Tornado Cash) are likely malicious.
Activities	Contracts that frequently interact with users exhibiting diverse behaviors are likely benign.
Source Code	Contracts with unverified source code are likely malicious.

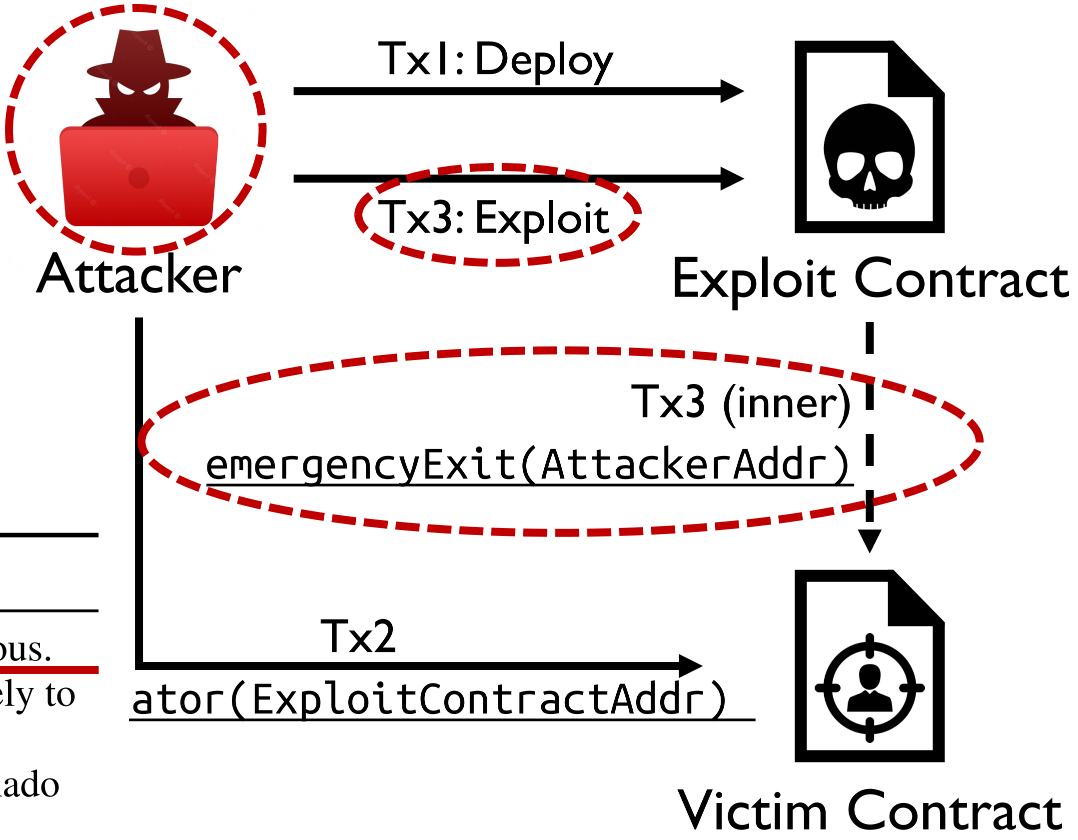


Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

- The attacking transaction is detected based on the profit.



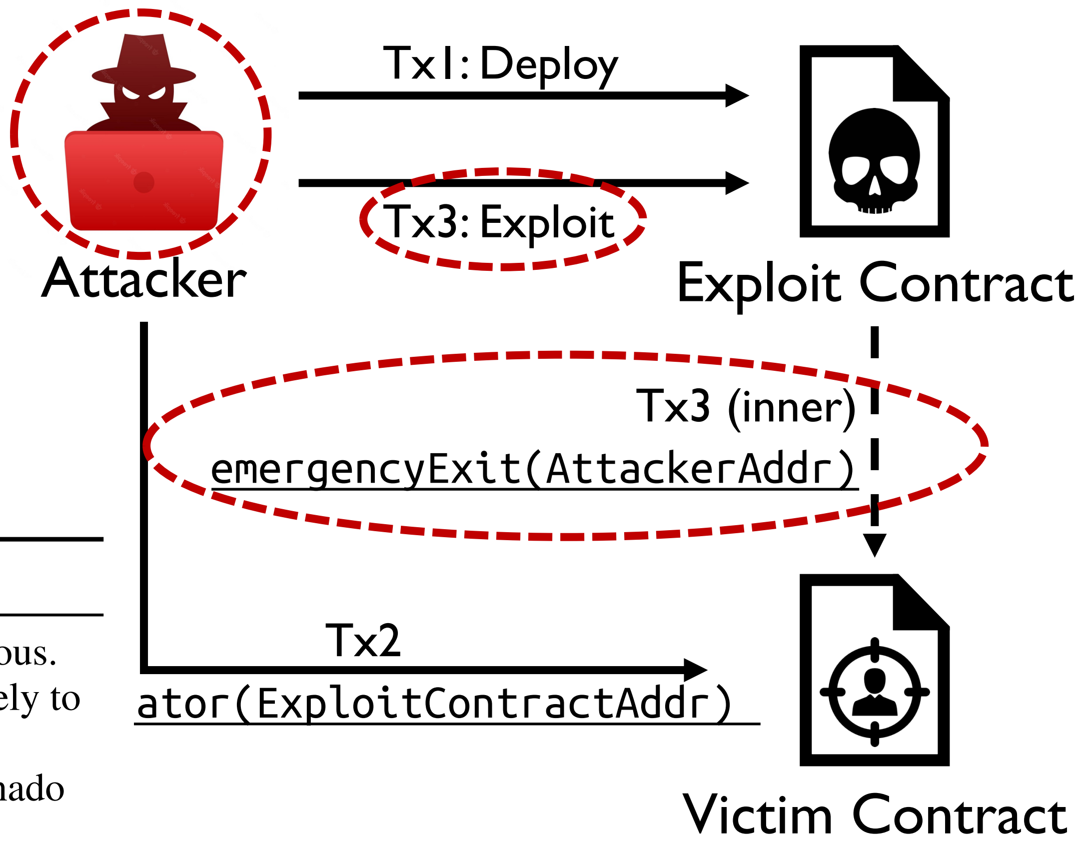
Name	Description
Lifespan	Contracts deployed shortly before an attack are likely malicious.
Balance	Contracts whose initial assets exceed the attack profit are likely to be victims.
Fund Source	Contracts and Wallets funded from mixing servers (e.g., Tornado Cash) are likely malicious.
Activities	Contracts that frequently interact with users exhibiting diverse behaviors are likely benign.
Source Code	Contracts with unverified source code are likely malicious.

Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

- The attacking transaction is detected based on the profit.



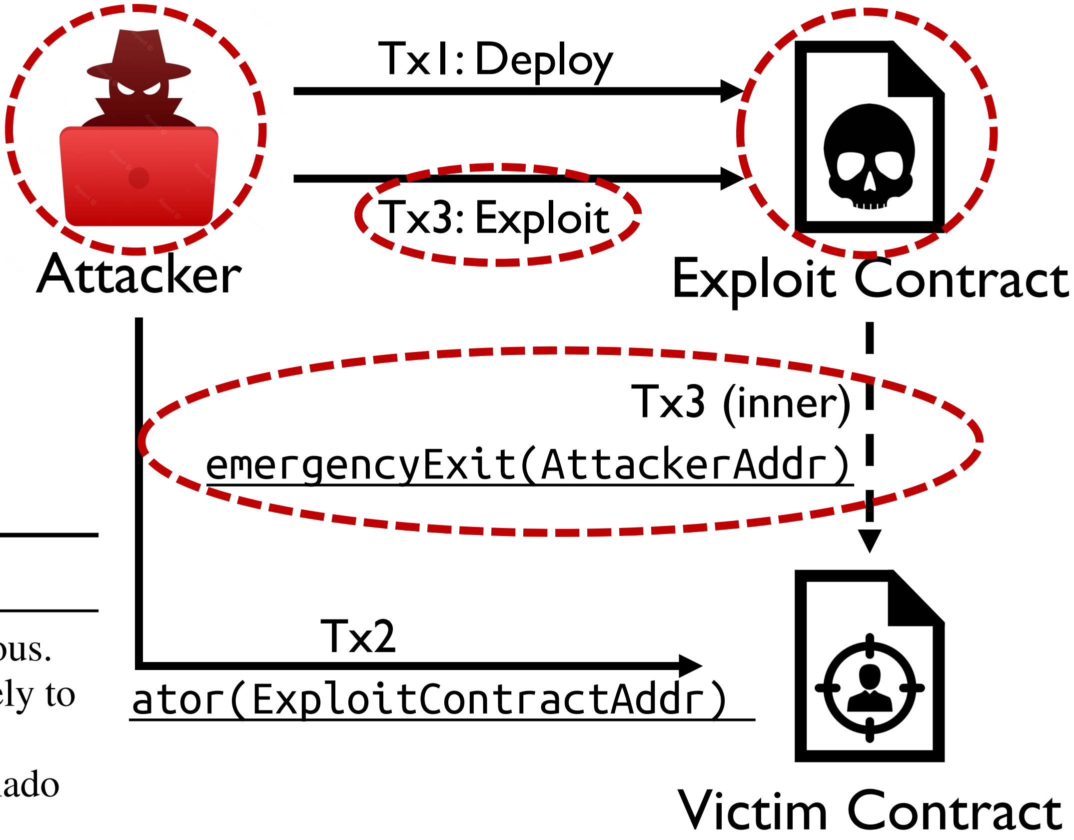
Name	Description
Lifespan	Contracts deployed shortly before an attack are likely malicious.
Balance	Contracts whose initial assets exceed the attack profit are likely to be victims.
Fund Source	Contracts and Wallets funded from mixing servers (e.g., Tornado Cash) are likely malicious.
Activities	Contracts that frequently interact with users exhibiting diverse behaviors are likely benign.
Source Code	Contracts with unverified source code are likely malicious.

Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

- The attacking transaction is detected based on the profit.



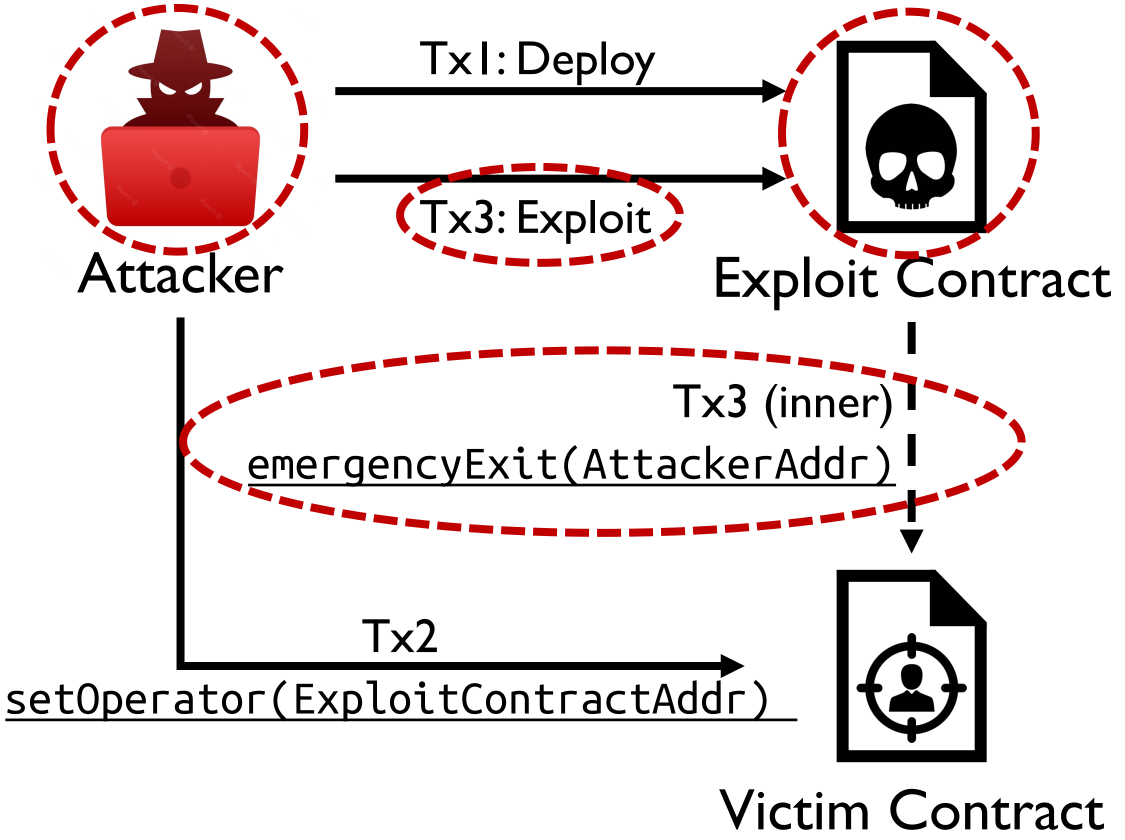
Name	Description
Lifespan	Contracts deployed shortly before an attack are likely malicious.
Balance	Contracts whose initial assets exceed the attack profit are likely to be victims.
Fund Source	Contracts and Wallets funded from mixing servers (e.g., Tornado Cash) are likely malicious.
Activities	Contracts that frequently interact with users exhibiting diverse behaviors are likely benign.
Source Code	Contracts with unverified source code are likely malicious.

Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

- The attacking transaction is detected based on the profit.
- Accounts are pinpointed based on historical behaviors.
- Transactions are pinpointed by read-write dependency.

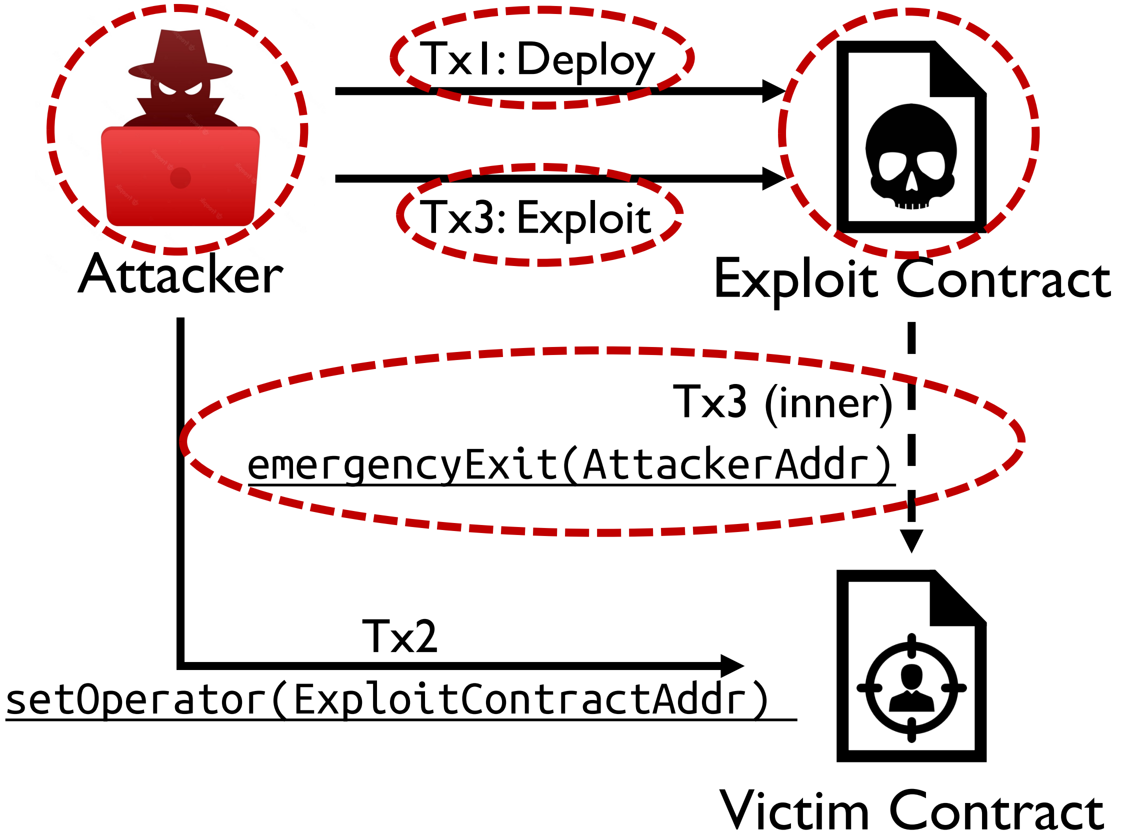


Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

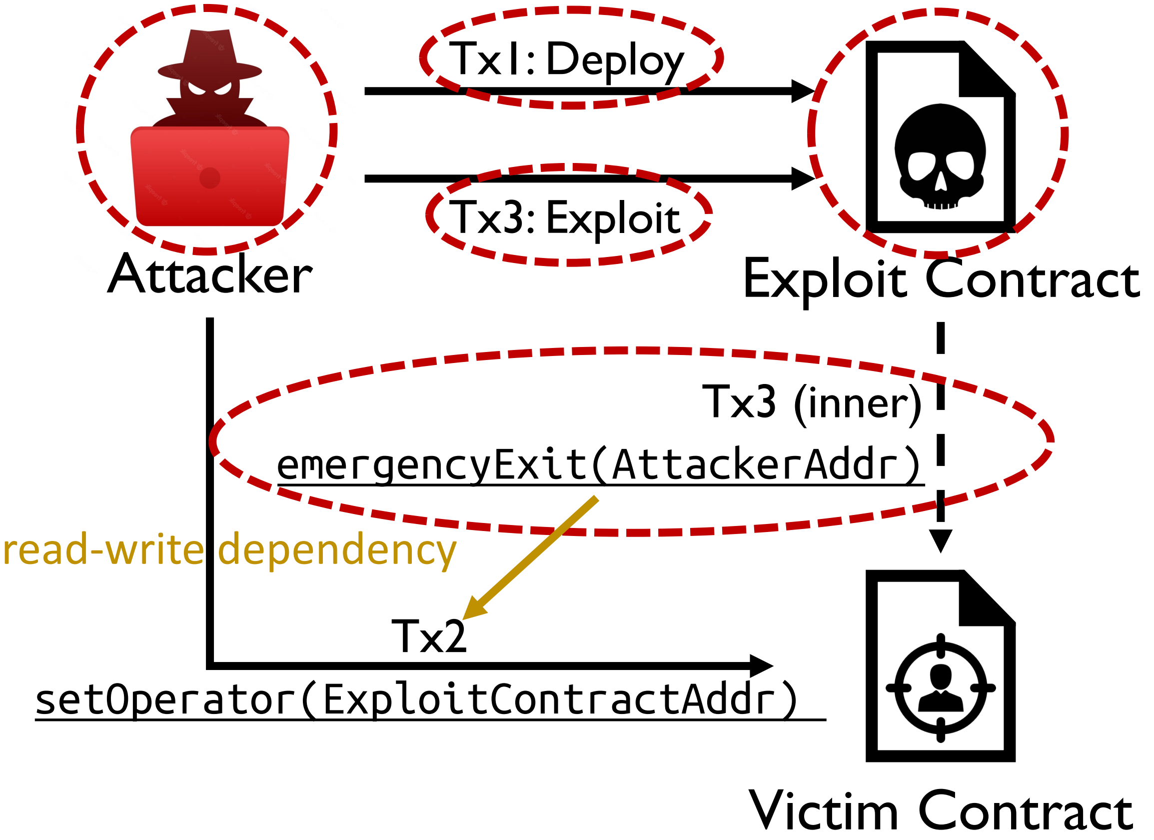
- The attacking transaction is detected based on the profit.
- Accounts are pinpointed based on historical behaviors.
- Transactions are pinpointed by read-write dependency.



Attack Information Identification

Vulnerable Contract

```
contract Victim {
  address operator;

  function setOperator(address _operator) {
    operator = _operator;
  }

  function emergencyExit(address to) {
    require(operator == msg.sender);
    to.transfer(address(this).balance);
  }
}
```

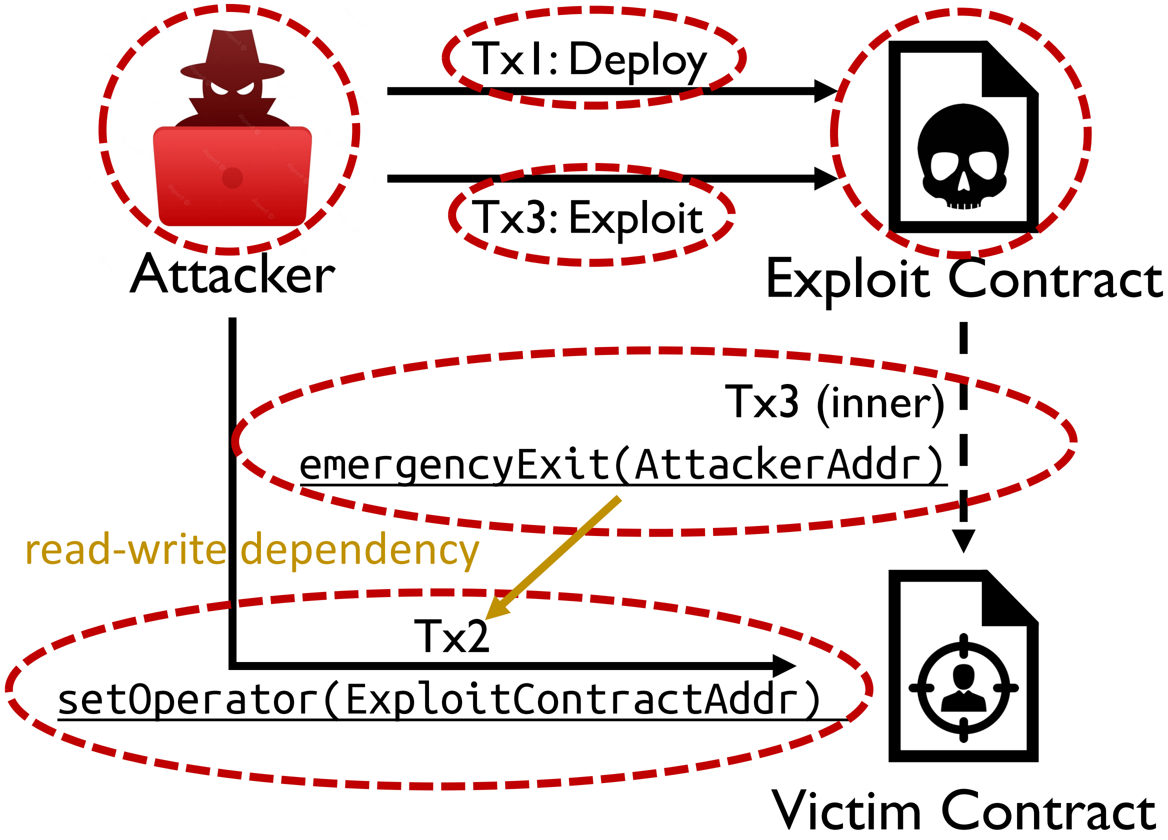


Attack Information Identification

Goal: Pinpoint all attack-related malicious entities, including accounts and transactions.

Transactions and accounts are pinpointed in an iterative fashion.

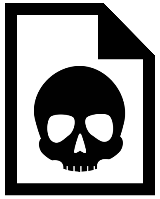
- The attacking transaction is detected based on the profit.
- Accounts are pinpointed based on historical behaviors.
- Transactions are pinpointed by read-write dependency.



Counterattack Smart Contract Synthesis

Goal: For each exploit contract, we aim to synthesize a counterattack contract that ensures the stolen funds are sent to accounts under our control.

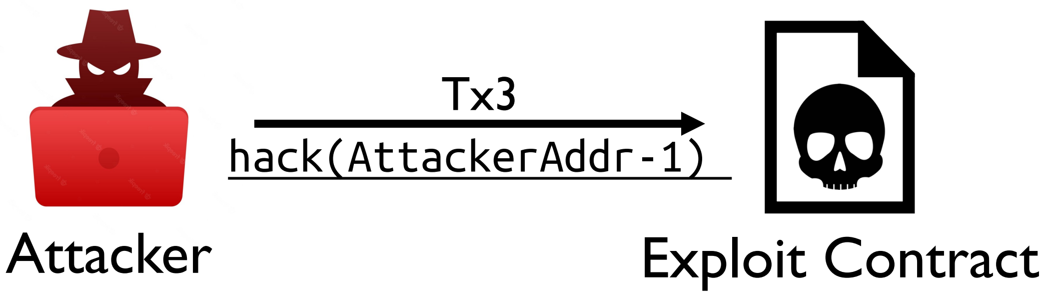
Attacker



Exploit Contract

Counterattack Smart Contract Synthesis

Goal: For each exploit contract, we aim to synthesize a counterattack contract that ensures the stolen funds are sent to accounts under our control.

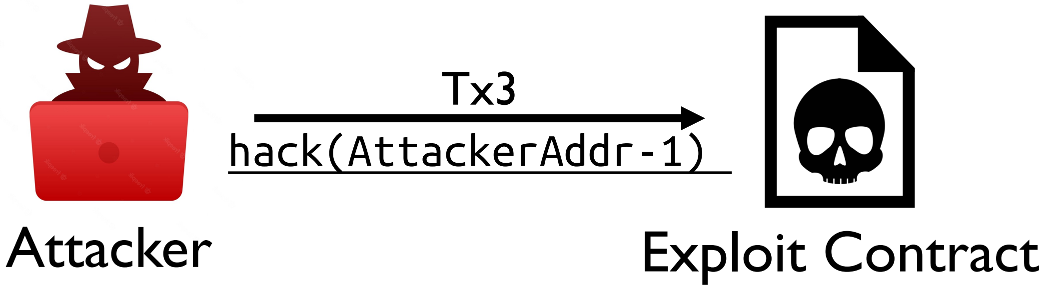


Exploit Contract

```
contract Exploit {
  function hack(address toAddr) {
    if (msg.sender == AttackerAddr) {
      VICTIM.emergencyExit(toAddr + 1);
    }
  }
}
```

Counterattack Smart Contract Synthesis

Goal: For each exploit contract, we aim to synthesize a counterattack contract that ensures the stolen funds are sent to accounts under our control.



Exploit Contract

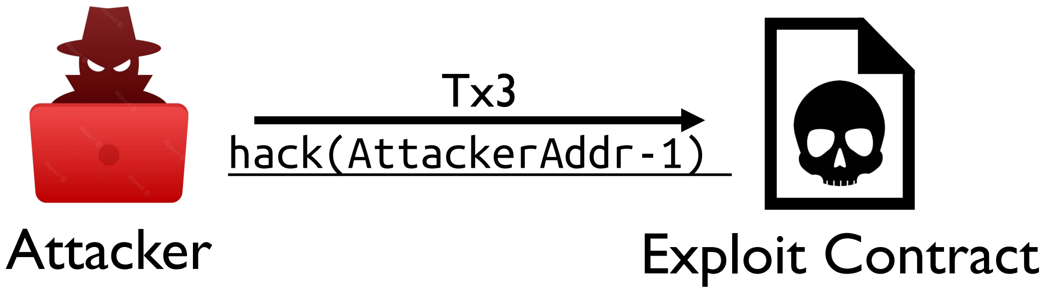
```
contract Exploit {
  function hack(address toAddr) {
    if (msg.sender == AttackerAddr) {
      VICTIM.emergencyExit(toAddr + 1);
    }
  }
}
```

Counterattack Contract

```
contract Exploit {
  function hack(address toAddr) {



  }
}
```

Counterattack Smart Contract Synthesis

Goal: For each exploit contract, we aim to synthesize a counterattack contract that ensures the stolen funds are sent to accounts under our control.



Exploit Contract

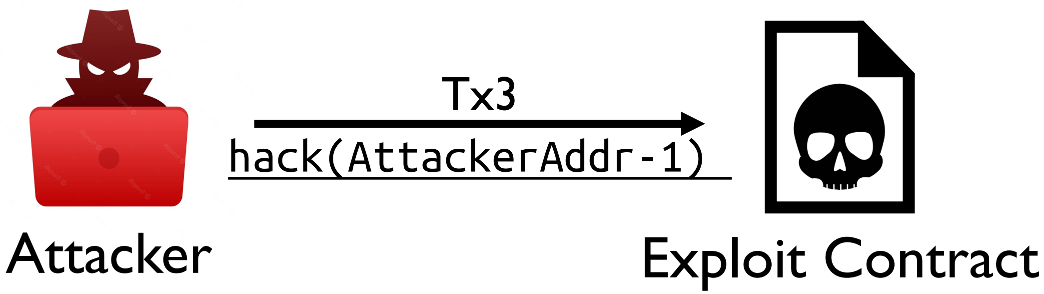
```
contract Exploit {
  function hack(address toAddr) {
    if (msg.sender == AttackerAddr) {
      VICTIM.emergencyExit(toAddr + 1);
    }
  }
}
```

Counterattack Contract

```
contract Exploit {
  function hack(address toAddr) {



  }
}
```

Counterattack Smart Contract Synthesis

Goal: For each exploit contract, we aim to synthesize a counterattack contract that ensures the stolen funds are sent to accounts under our control.



Exploit Contract

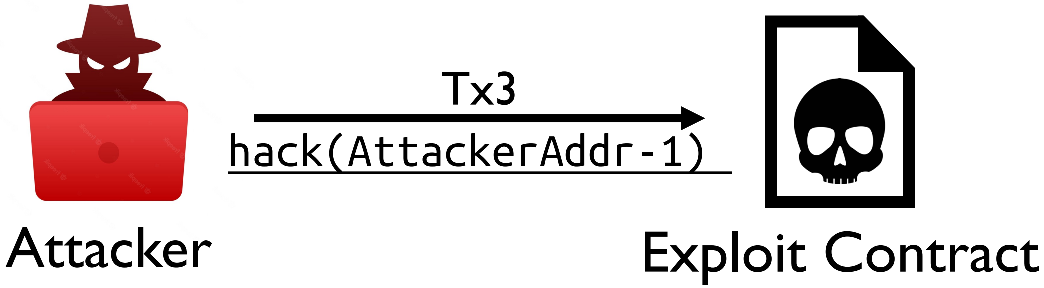
```
contract Exploit {
  function hack(address toAddr) {
    if (msg.sender == AttackerAddr) {
      VICTIM.emergencyExit(toAddr + 1);
    }
  }
}
```

Counterattack Contract

```
contract Exploit {
  function hack(address toAddr) {
    if (true) {

    }
  }
}
```

Counterattack Smart Contract Synthesis

Goal: For each exploit contract, we aim to synthesize a counterattack contract that ensures the stolen funds are sent to accounts under our control.



Exploit Contract

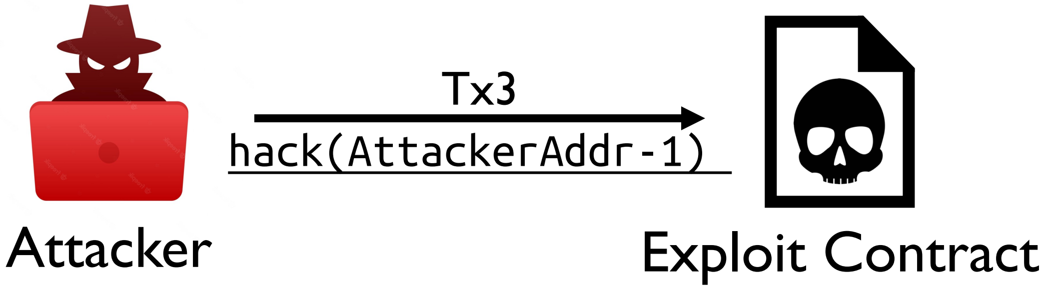
```
contract Exploit {
  function hack(address toAddr) {
    if (msg.sender == AttackerAddr) {
      VICTIM.emergencyExit(toAddr + 1);
    }
  }
}
```

Counterattack Contract

```
contract Exploit {
  function hack(address toAddr) {
    if (true) {

    }
  }
}
```

Counterattack Smart Contract Synthesis

Goal: For each exploit contract, we aim to synthesize a counterattack contract that ensures the stolen funds are sent to accounts under our control.



Exploit Contract

```
contract Exploit {
  function hack(address toAddr) {
    if (msg.sender == AttackerAddr) {
      VICTIM.emergencyExit(toAddr + 1);
    }
  }
}
```

Counterattack Contract

```
contract Exploit {
  function hack(address toAddr) {
    if (true) {
      VICTIM.emergencyExit(OurAddress);
    }
  }
}
```

Contract Execution and Validation

Goal: Ensure the success of the counterattack by locally deploying the synthesized contract, guaranteeing that it will result in a profit to our addresses.

Evaluation

STING

Evaluation

Dataset:

We investigated a total of **86** attacks that occurred on the Ethereum mainnet prior to 2023, of which **24** are deemed out of scope.

Attack	Date	Loss	Root Cause
Wintermute	09/20/22	160.0M	Key compromised or rugged
SudoRare	08/23/22	800.0K	Key compromised or rugged
Curve Finance	08/09/22	575.0K	Off-chain component compromise
Harmony Bridge	06/24/22	100.0M	Key compromised or rugged
Ronin Network	03/29/22	624.0M	Key compromised or rugged
BuildFinance	02/14/22	470.0K	Key compromised or rugged
Dego Finance	02/10/22	10.0M	Key compromised or rugged
Meter	02/06/22	7.7M	No fund lost on the mainnet
Qubit Finance	01/28/22	80.0M	No fund lost on the mainnet
Crypto.com	01/18/22	33.7M	Key compromised or rugged
LCX	01/08/22	7.9M	Key compromised or rugged
Vulcan Forged	12/13/21	140.0M	Key compromised or rugged
Bitmart	12/04/21	196.0M	Key compromised or rugged
Badger	12/02/21	120.0M	Off-chain component compromise
AnubisDAO	10/29/21	60.0M	Key compromised or rugged
JayPegs Automart	09/17/21	3.1M	Key compromised or rugged
DAO Maker	08/12/21	7.0M	Key compromised or rugged
Thorchain	07/22/21	8.0M	Off-chain component compromise
Thorchain	07/15/21	5.0M	Off-chain component compromise
Bondly	07/15/21	5.9M	Key compromised or rugged
Anyswap	07/10/21	7.9M	Key compromised or rugged
Chainswap	07/02/21	800.0K	No fund lost on the mainnet
Roll	03/14/21	5.7M	Key compromised or rugged
Paid Network	03/05/21	3.0M	Key compromised or rugged

Evaluation

Dataset:

We investigated a total of **86** attacks that occurred on the Ethereum mainnet prior to 2023, of which **24** are deemed out of scope.

Effectiveness:

We successfully synthesized **54** counterattacks out of **62** attacks.

Evaluation

Dataset:

We investigated a total of **86** attacks that occurred on the Ethereum mainnet prior to 2023, of which **24** are deemed out of scope.

Effectiveness:

We successfully synthesized **54** counterattacks out of **62** attacks.

Efficiency:

The median runtime overhead is **0.29** seconds, while the worst-case value rises to **8.51** seconds (only two cases exceed **1.00** second).

STING

Limitations

- Adaptive Evasion: Multiple adaptive evasion techniques, such as code obfuscation, may exist against STING, enabling attacks to circumvent our defense mechanism.
- Private Transactions: Private transactions provide a mechanism for blockchain users to execute transactions that remain hidden until being confirmed.
- Blind Spots: STING does not provide comprehensive protection against all DeFi attacks.
- Performance Issue: The execution overhead of STING is not optimal for MEV bots to initiate front-running transactions, with a worst-case duration of 8.51 seconds.
  - Our prototype implementation is not optimal: **3.3** seconds (for our archive node) vs. **0.74** seconds (for Reth) in the 8.51-second worst-case scenario.

Related Works

Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? In 2022 IEEE Symposium on Security and Privacy (SP), pages 198–214. IEEE, 2022.

Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the defi ecosystem with flash loans for fun and profit. In Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I, pages 3–32. Springer, 2021.

Liyi Zhou, Kaihua Qin, Antoine Cully, Benjamin Livshits, and Arthur Gervais. On the just-in-time discovery of profit-generating transactions in defi protocols. In 2021 IEEE Symposium on Security and Privacy (SP), pages 919–936. IEEE, 2021.

Dabao Wang, Siwei Wu, Ziling Lin, Lei Wu, Xingliang Yuan, Yajin Zhou, Haoyu Wang, and Kui Ren. Towards a first step to understand flash loan and its applications in defi ecosystem. In Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing, pages 23–28, 2021.

Homepage

Thank You

Zhuo Zhang, zhan3299@purdue.edu

PURDUE UNIVERSITY

THE OHIO STATE UNIVERSITY

August 9, 2023