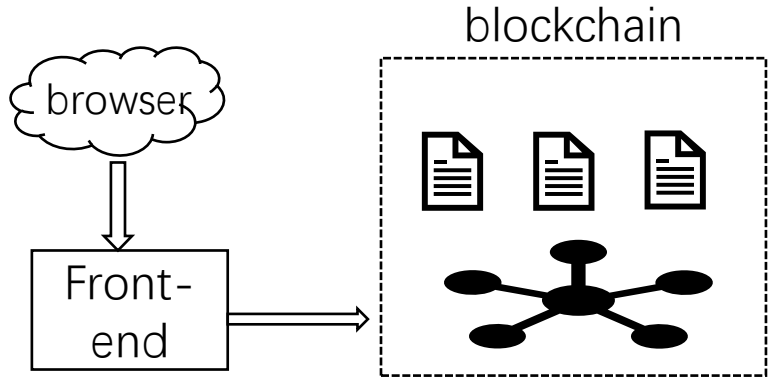
Evil Under the Sun: Understanding and Discovering Attacks on Ethereum Decentralized Applications

Liya Su[1,2,3] *†, Xinyue Shen[1,4]*†, Xiangyu Du[1,2,3]*, Xiaojing Liao[1],
XiaoFeng Wang[1], Luyi Xing[1], Baoxu Liu[2,3]
[1]*Indiana University Bloomington,* [2]*Institute of Information Engineering, Chinese Academy of Sciences,*
[3]*University of Chinese Academy of Sciences,* [4]*Alibaba Group*
{liyasu, shen12, duxian}@iu.edu, {xliao, xw7, luyixing}@indiana.edu, liubaoxu@iie.ac.cn

USENIX Security 2021

Background



- Decentralized applications(Dapp)

1,169 Dapps with 5,786 contract addresses and 18 categories

（from DAPP list ）

- Two types of accounts:

（1） Externally Owned Accounts (EOAs)

（2） Contract Accounts （smart contracts）

Difference ： w3.eth.getCode()

Background

- **Transaction**:a signed data package storing a message

`From(R,S,V)`: Sender's signature

`To`: the recipient(The 160-bit address)

`Value`: the amount of money transferred from the sender to the recipient

`Data`: the input for a contract

`gasprice`

`gas`

TO	0x54*
FROM	0x73*
VALUE	0.01 Ether
DATA	0xc52ab778 (methodID of function execute())
GAS PRICE	6.3×10^{-9} Ether (6.3 Gwei)

Problem&Challenges

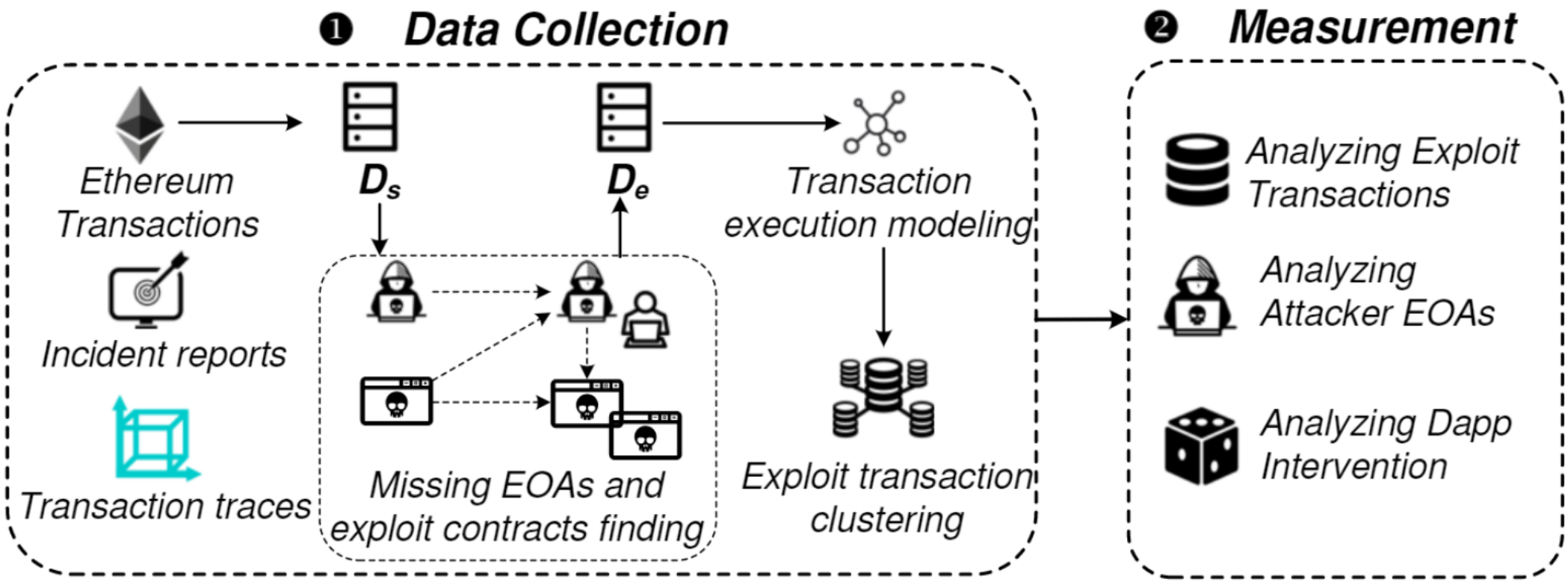
- What like and how the attacks launch on Dapps?

— Mainly for the back-end

- How to automatically reconstruct Dapp attacks

- How to find new attack and prevent it

Framework

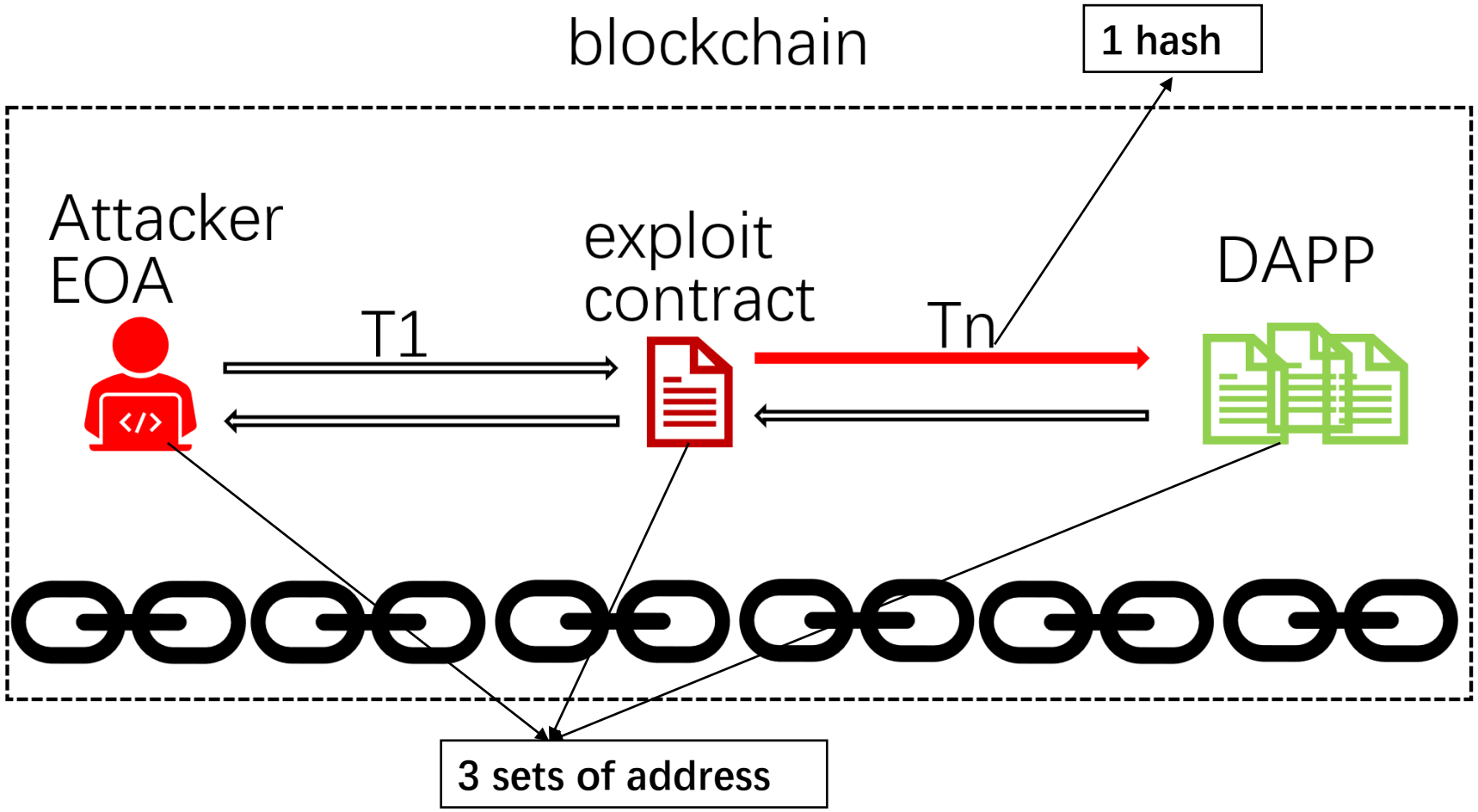

❶ Data Collection
❷ Measurement
Ethereum Transactions
Incident reports
Transaction traces
D_s
D_e
Missing EOAs and exploit contracts finding
Transaction execution modeling
Exploit transaction clustering
Analyzing Exploit Transactions
Analyzing Attacker EOAs
Analyzing Dapp Intervention

- Data Collection and Derivation

To Collect all transaction information related to the attack

1：Build seed attack set **Ds(3 sets of addresses +1 hash)** from Internet

2：Reconstruct the reported incident



• Data Collection and Derivation

To expand the seed attack set Ds

Step1: Get more EOA from transactions.

Check transactions related to EOA or exploit contract

TO	0x54*
FROM	0x73*
VALUE	0.01 Ether
DATA	0xc52ab778 (methodID of function execute())
GAS PRICE	6.3×10^{-9} Ether (6.3 Gwei)

Step2: Get more similar exploit contract.

(1)Get all related contracts within a timewindow (1 day)

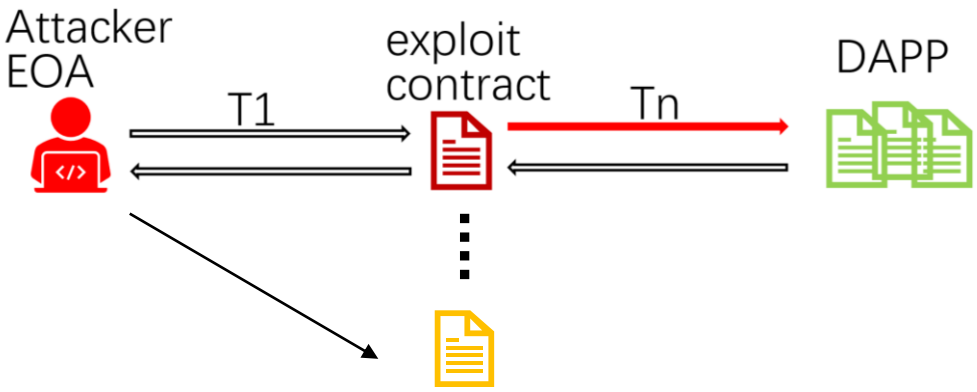
(2) compare similarities with opcode Jaccard similarity

(3)If Jaccard similarity ⩾ 0.9;Add it.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Get new set De

Ds to De

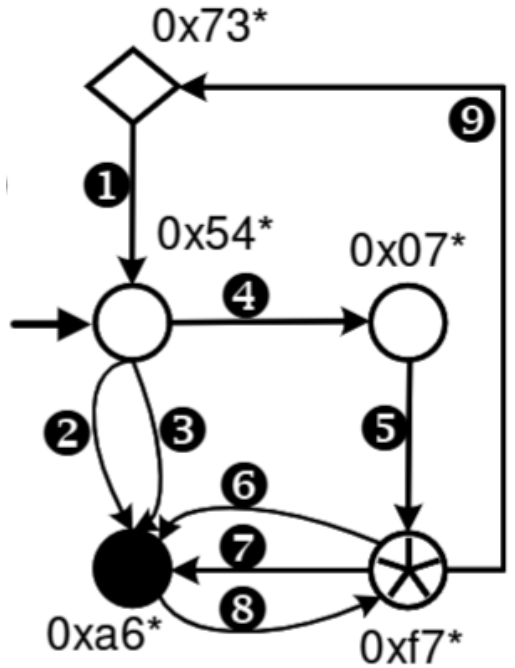


- **Exploit transaction clustering**

(1) Transaction execution modeling

Model execution traces $\boldsymbol{e_t}$ (Ii, Oi, Bi, Ti)

={from,to,function,value}



(2) A graph **T G** = (V, E, W,t)

(3) D(g1, g2) = distance(similarity and time)

(4) Clustering with k-means algorithm(126 clusters)(42 Dapp attack incidents with 58,555 transactions)

- Attack footprints

A typical Dapp attack has 4 stages

Stage 1. Preparation: Related transactions before Exploitation

Purposes :

(1) Testing exploit contracts → **Exploit Developer**

(2) Prepare start-up funds → **Money Manager**

85% of attack incidents with the average number of transactions being 23

- Attack footprints

Stage 2. Exploitation:
when the attacker continuously makes profits from one Dapp

transaction

TO	0x54*
FROM	0x73*
VALUE	0.01 Ether
DATA	0xc52ab778 (methodID of function execute())
GAS PRICE	6.3×10^{-9} Ether (6.3 Gwei)

Purposes :
(1) invoke vulnerable Dapp functions
(2) deploy or trigger an exploit contract to automate an attack

More:Attacker tends to rapidly evolve his strategies via delegatecall(), or creating new contracts(1,394transactions from 6 attacker EOAs)

- Attack footprints

Stage 3. Propagation:

creating a new contract and reuse the exploit(4 more DAPPs)

Purposes :

(1) Get more profits → **Attack Operotor**

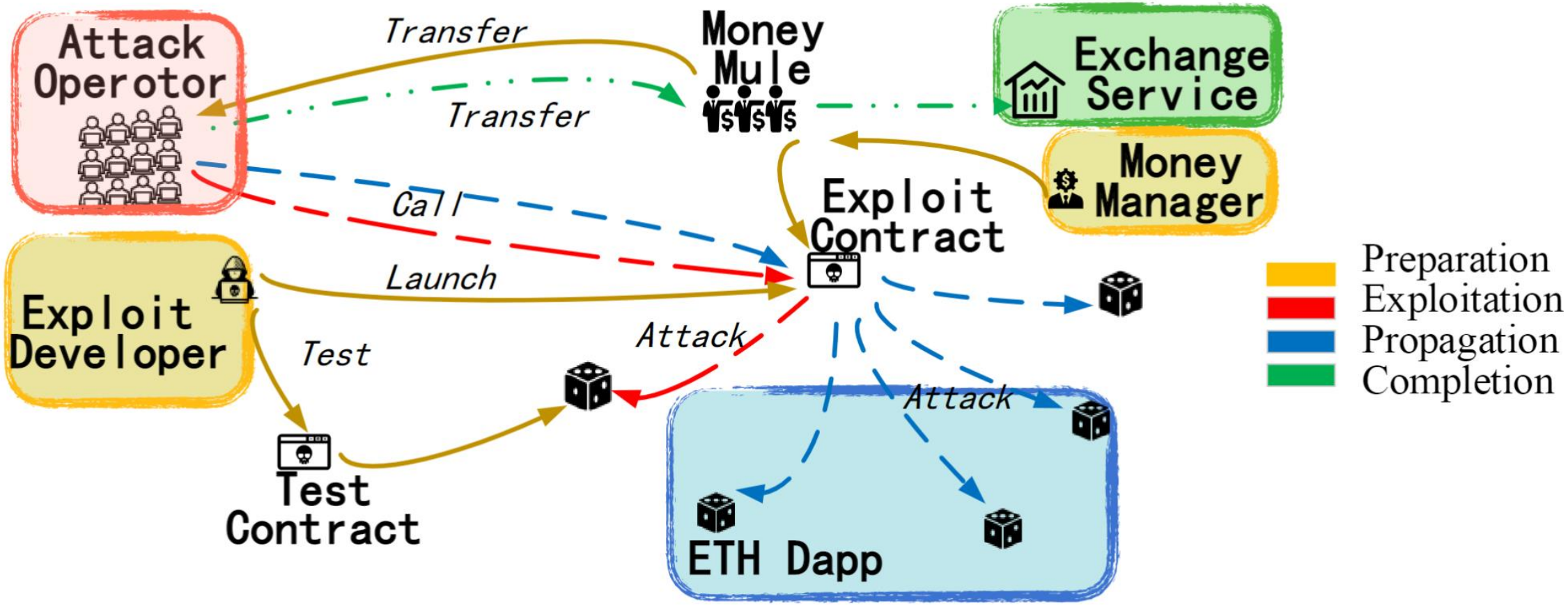
Stage 4. Mission completion:

Puposes:

Remove attack traces and get profits → **Money Mule**

selfdestruct() and transfer money

- Attack footprints



- Finding New Attacks

Key insights: highlevel behavior patterns are relatively stable in each attack stage

Tool:**DEFIER(2 parts)**

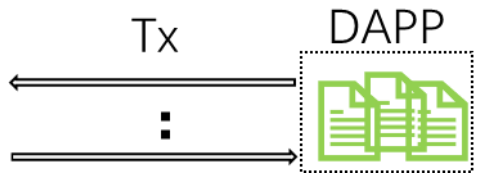
(1)Preprocessing （Ds to De）

Input : transactions with a DAPP

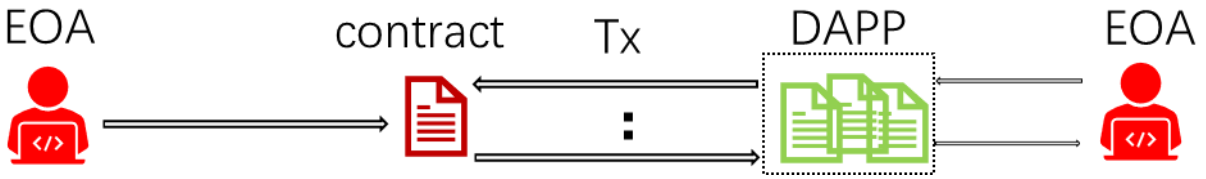
Output : transactions groups

- Finding New Attacks

Step1:get transaction



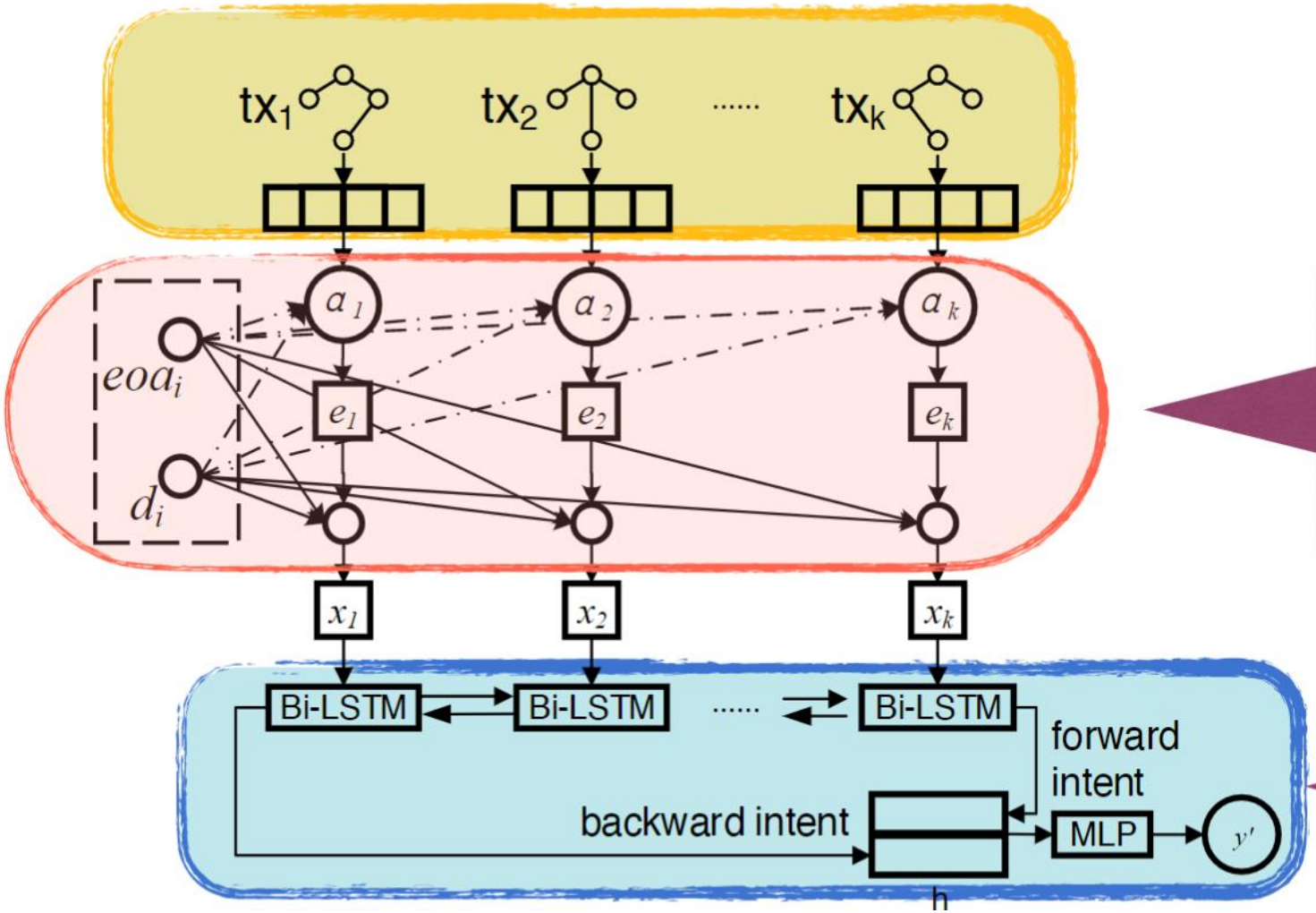
Step2:get EOAs+contract



Step3:get similar Tx

- Finding New Attacks

(2) Sequence-based Classification



EOA-Dapp-execution attention model: highlight the useful information related to the EOA's intent on the Dapp.

Output types: normal, preparation, exploitation, propagation and completion.

Evaluation

- Measurement

Table : Known Dapp attacks

Attack type	# of Dapps		# of exploit contracts		# of attacker EOAs		# of attack transactions	
	D_s	D_e	D_s	D_e	D_s	D_e	D_s	D_e
Bad randomness	4	14	9	19	9	27	14	40,766
DoS	4	6	3	3	5	88	4	17,088
Integer overflow/underflow	13	32	1	2	28	53	47	591
Reentrancy	2	2	2	3	2	4	2	30
Improper authentication	12	18	6	18	17	60	34	575
Unique total	25	56	20	45	48	227	77	58,555

Table : List of vulnerable functions

Functions	#Dapp	Attack type
transferFrom	16	Integer overflow/underflow
airDrop	8	Bad randomness
transfer	7	Integer overflow/underflow
transferProxy	6	Integer overflow/underflow
batchTransfer	5	Integer overflow/underflow

Role overlap of attacker EOAs

Evaluation

- Evaluation with groundtruth set

469, 22333
34763, 290

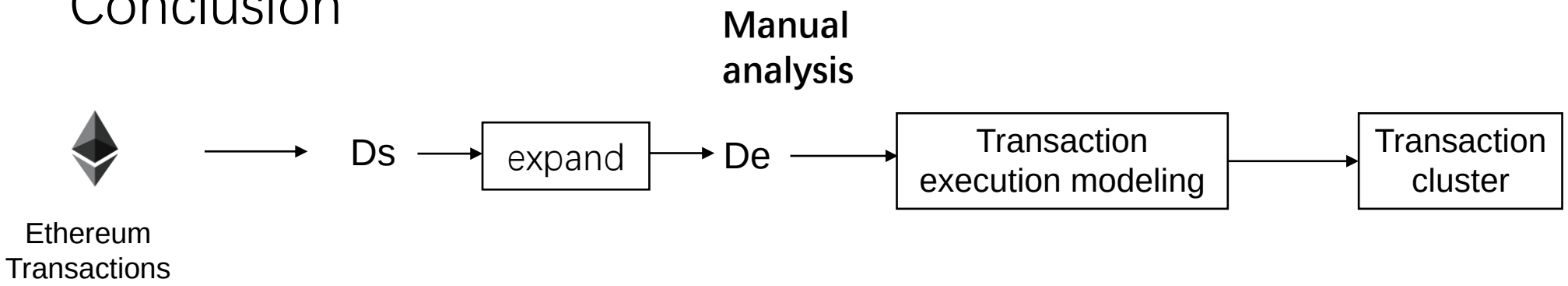
Dataset	# transactions	Results
Groundtruth set	badset 57,855 goodset 39,124	pre_{micro} 98.2%, pre_{macro} 92.4% rec_{micro} 98.1%, rec_{macro} 98.4%
Unknown set	2,350,779	*positive* 476,334
Sampled testset	30,888	pre_{micro} 91.7% pre_{macro} 83.6%

transactions that labeled as one of attack stages

Performance comparison in different models

Method	Attention	precision	recall	F1
RNN	no attention	0.965	0.962	0.963
RNN	attention	0.974	0.969	0.971
LSTM	no attention	0.977	0.975	0.976
LSTM	attention	0.982	0.981	0.981

Conclusion



→ *Cyber threat intelligence* (CTI) 4 stages

Tool: DEFIER