THE ART OF THE SCAM: DEMYSTIFYING HONEYPOTS IN ETHEREUM SMART CONTRACTS

Presented by: Al Shafayet, Syam Badri

We will be covering

01

Ethereum Smart Contracts

02

Honeypots: Attackers View

03

Honeypot detection System

Ethereum

Decentralized platform that allows developers to build applications using blockchain

Smart Contracts

- Self-executing
- Stored in blockchain
- Immutable, guaranteed execution
- Highlevel language, eg: Solidity
- Identified as 160 bit address
- Can be transferred or destroyed

Ethereum Virtual Machine(EVM)

- Transaction based state machine
- Decentralized and isolated
- Stack-based, register-less
- Runs low-level bytecode
- Uses “ether” as gas

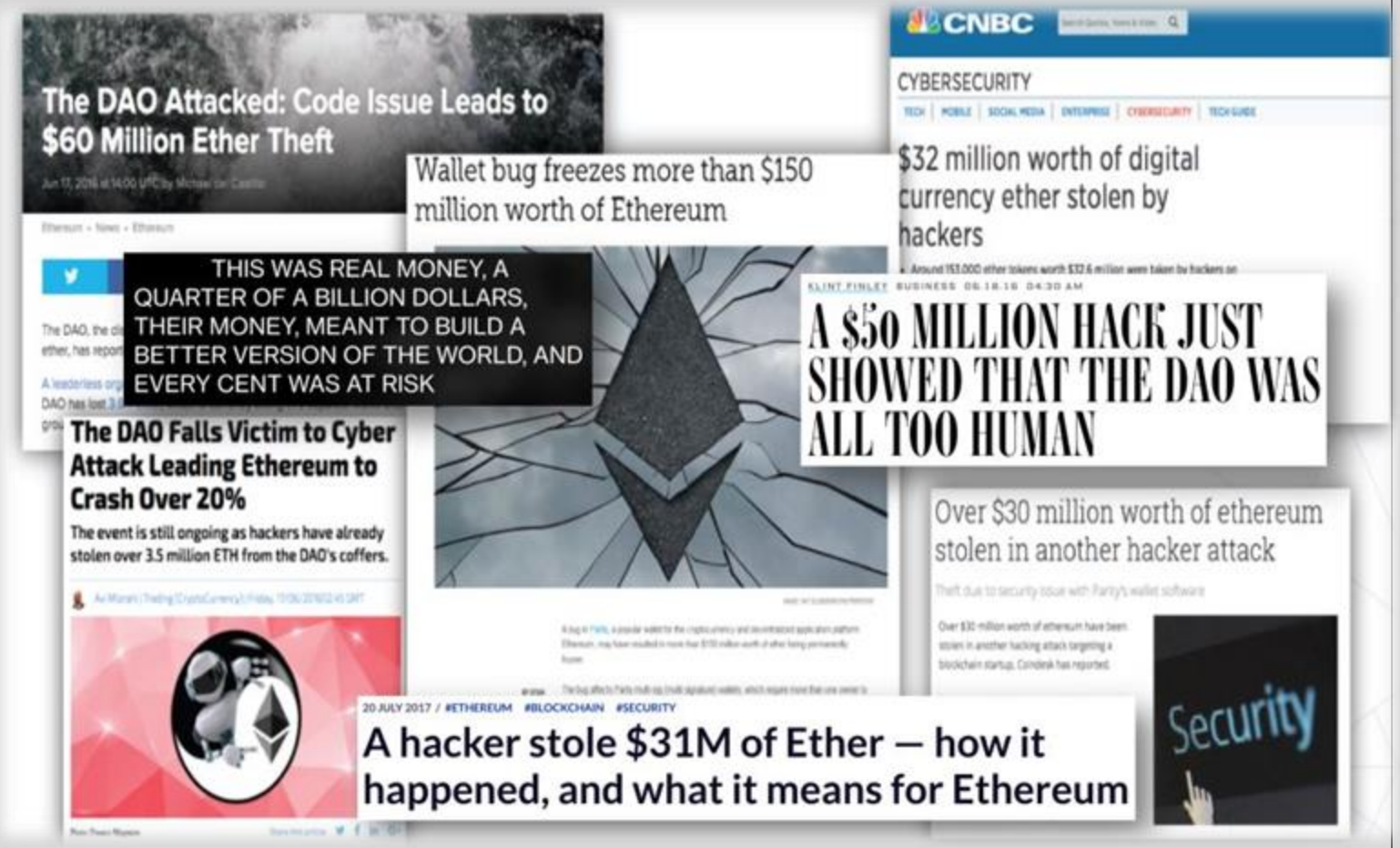
Attacks on smart contract

"In 2016 DAO attack drained over 50 million worth ether."

- Exploits vulnerability
- Reentracy Attack: contract repeatedly calls itself
- Integer Overflow: variable overflows

To prevent contract vulnerability:

Verification, Audits, Bug bounty Programs, Code Standards



Honeypots

Smart Contracts **pretending** to be vulnerable to attract users, but actually exploits the user.

Why should I spend time looking for victims, if I can let the victims come to me!!

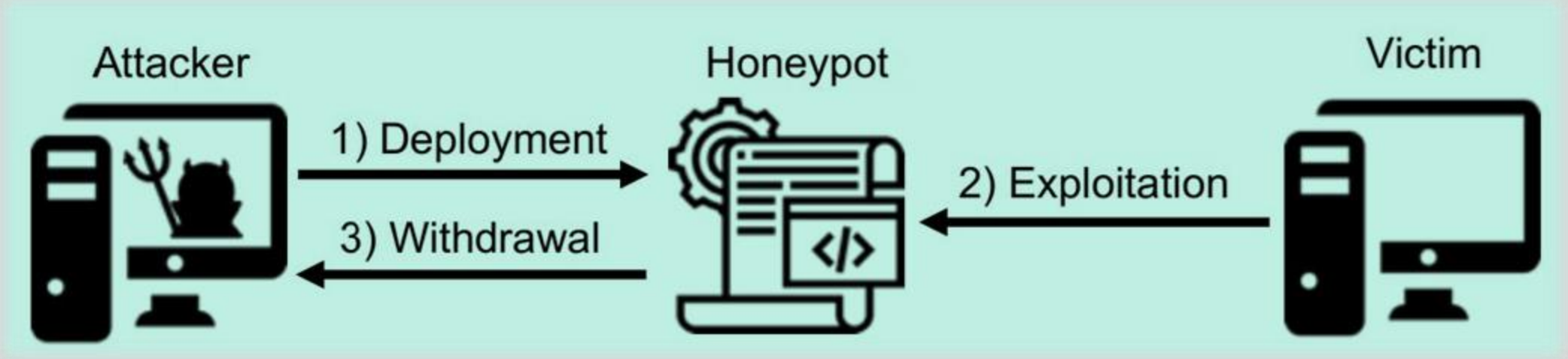
```
Contract multiplicatorX3{
        address public Owner=msg.sender;
        function() public payable{}
        function withdraw()
        payable
        public
        {
                require(msg.sender==Owner);
                Owner.transfer(this.balance);}
```

Trap

Bait

```
function Command(address adr, bytes, data)
payable
public
{
        require(msg.sender==Owner);
        adr.call.value(msg.value)(data);}

function multiplicate(address adr)
public
payable
{       if(msg.value>=this.balance){
        adr.transfer(this.balance+msg.value);}}}
```



Level	Technique
Ethereum Virtual Machine	I. Balance Disorder
Solidity Compiler	I. Inheritance Disorder II. Skip empty string literal III. Type deduction overflow IV. Uninitialized struct
Etherscan Blockchain Explorer	I. Hidden State Update II. Hidden transfer III. Straw man contract

Honeypots Detection System

Takes bytecodes and returns detailed report of detected honeypots.

Symbolic Analysis:

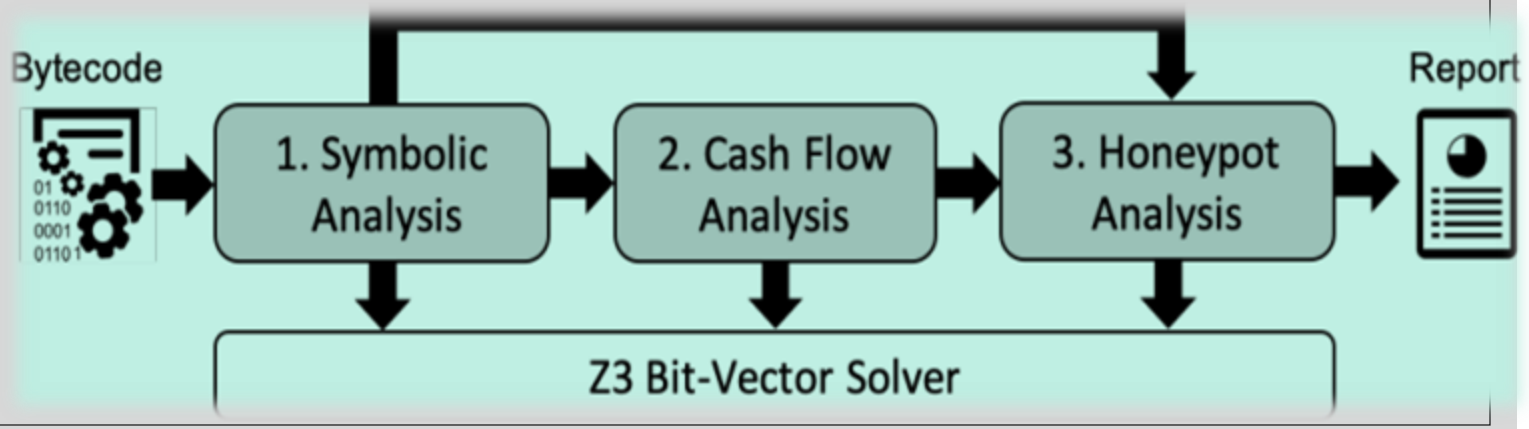
- Based on Luu et al's Oyente
- Collects meta information=>Storage write S, Execution Path P, Arithmetic Operation A etc.
- Creates mathematical model

Cash Flow Analysis:

- Discards contracts that does not consist of receive or transfer funds
- Traces movement of funds

Honeypot analysis:

- Detects honeypot techniques
- Can be extended on newly-found honeypots



Get the codes @github.com/christoftorres/HoneyBadger

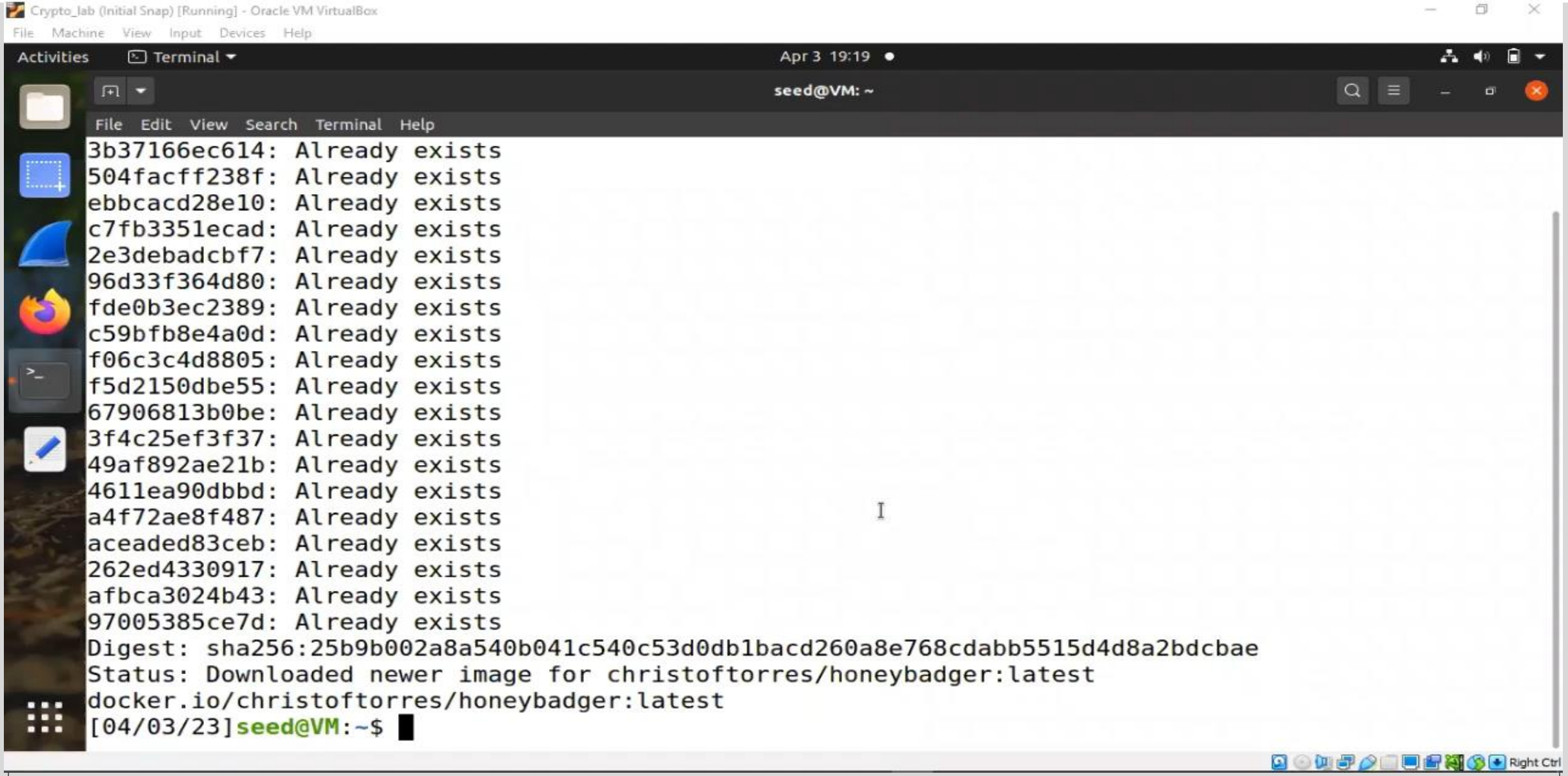
To implement “Honeybadger” on your system:

- ❑Download docker image-→ *docker pull christoftorres/honeybadger*
- ❑Run the honeybadger docker container→ *docker run -i -t christoftorres/honeybadger*

To evaluate honeypot simply run:

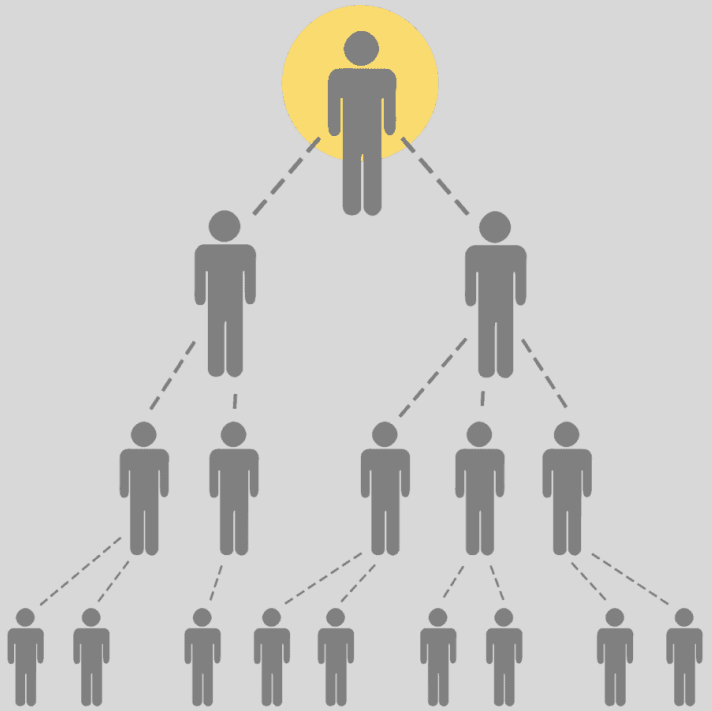
python honeybadger/honeybadger.py -s honeypots/MultiplicatorX3.sol

You can also get updated on Ethereum contracts that are malicious **@honeybadger.uni.lu**


Crypto_lab (Initial Snap) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Apr 3 19:19
seed@VM: ~
File Edit View Search Terminal Help
3b37166ec614: Already exists
504facff238f: Already exists
ebbcacd28e10: Already exists
c7fb3351ecad: Already exists
2e3debadcbf7: Already exists
96d33f364d80: Already exists
fde0b3ec2389: Already exists
c59bfb8e4a0d: Already exists
f06c3c4d8805: Already exists
f5d2150dbe55: Already exists
67906813b0be: Already exists
3f4c25ef3f37: Already exists
49af892ae21b: Already exists
4611ea90dbbd: Already exists
a4f72ae8f487: Already exists
aceaded83ceb: Already exists
262ed4330917: Already exists
afbca3024b43: Already exists
97005385ce7d: Already exists
Digest: sha256:25b9b002a8a540b041c540c53d0db1bacd260a8e768cdabb5515d4d8a2bdcbae
Status: Downloaded newer image for christoftorres/honeybadger:latest
docker.io/christoftorres/honeybadger:latest
[04/03/23]seed@VM:~$
Right Ctrl

Enhancement: Ponzi Scheme

- Smart contracts are based on **Investment** strategy
- Ponzi scheme is based on real life pyramid scam
- They attract investors through “high-yield” investment schemes



```
address lastDepositor;
uint public currentAmount;

function Ponzi(uint _startingAmount) public {
currentAmount = _startingAmount;
 }

function pay() public payable {
  require(msg.value == currentAmount);
  lastDepositor.transfer(msg.value);
   lastDepositor = msg.sender;

round.
   currentAmount =
nextAmount(currentAmount);
 }

 function nextAmount(uint amount) private
pure returns (uint) {
   return amount * 2;
```

lastDepositor: Keeps track of last depositor in the chain
currentAmount: Amount of ether that needs to be deposited to get into the "Investment"

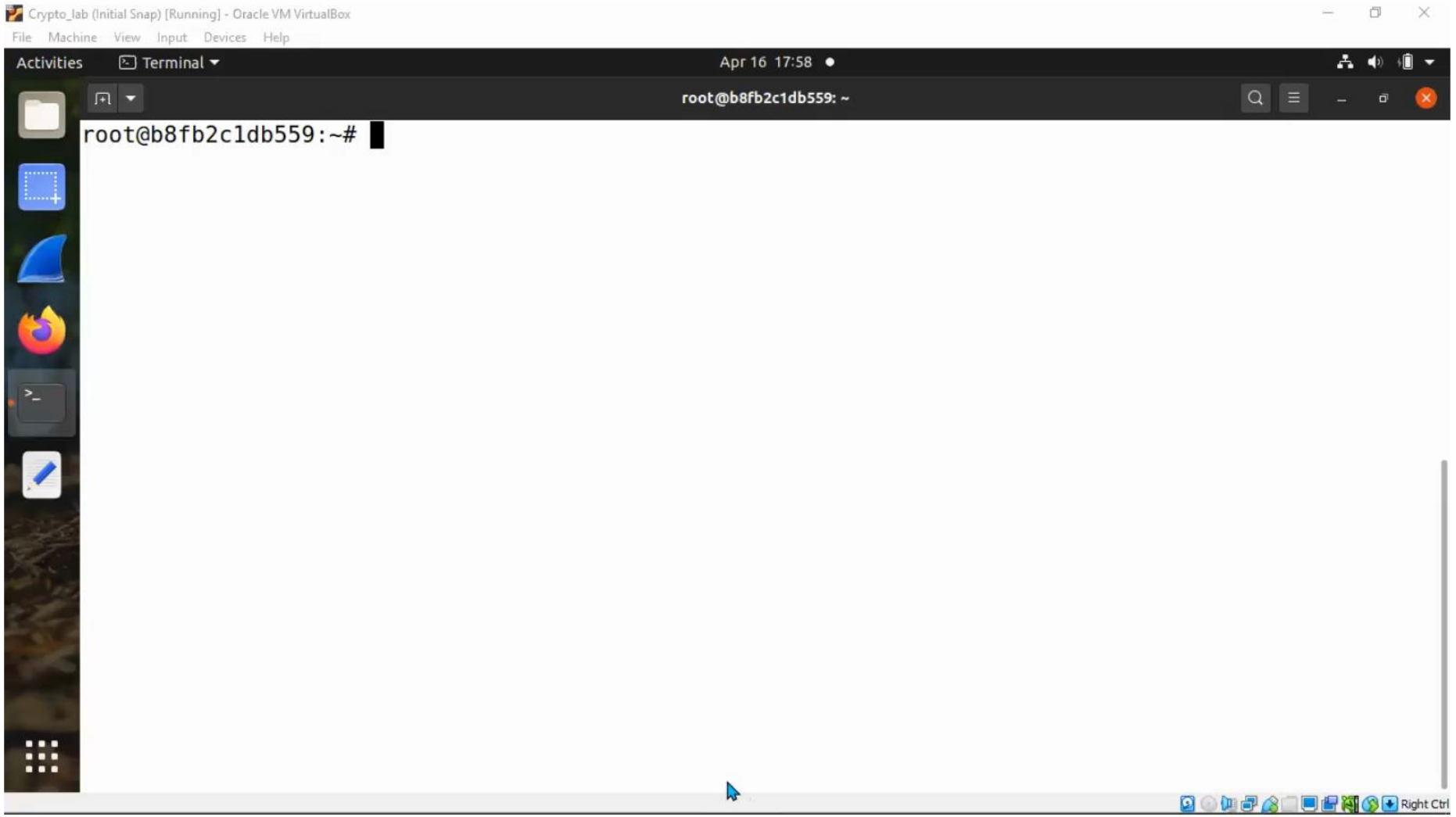
Users can deposit ETH and get interest; however, interest is only paid on deposits made by new users. The program will fail as soon as the number of new customers starts to diminish.

Symbolic Solution

Simplistic symbolic function:

- Follows address of receiver and transferrer
- Checks of potential interest fraud
- Looks at a simple pattern to detect ponzi schemes

https://raw.githubusercontent.com/silbunsa/crispy-octo-sniffle/main/multi.sol?token=GHSAT0AAAAAAB7E7PI4TERY33WOW2AVZB4OKEA


Crypto_lab (Initial Snap) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Apr 16 17:58
root@b8fb2c1db559: ~
root@b8fb2c1db559:~#
Right Ctrl

Thank you ACS 545

Feel free to ask us anything ! !