



**Instituto Tecnológico de Aeronáutica – ITA**

Divisão de Ciência da Computação

*Grupo de Pesquisa em Engenharia de Software – GPES/ITA*

**WAIAF 2019**

**Workshop of Artificial Intelligence Applied to Finance**

**Wibx: Making Smart Contracts Even Smarter**

**Coordenação Técnica**

Prof. Dr. Adilson Marques da CUNHA

**Scrum Master / Product Owner**

GILDÁRCIO Gonçalves

Prof. Dr. Luiz A. V. DIAS

**Pesquisador Responsável**

Rafael SHIGEMURA

São José dos Campos, Maio de 2019

# AGENDA

1. Introdução
2. Blockchains baseadas em Ethereum
3. *Smart Contracts* (SCs) e suas vulnerabilidades
4. Detecção automática de vulnerabilidades em SCs
5. Conclusão
6. Trabalhos Futuros

# 1. INTRODUÇÃO

*“Security can be no stronger than its weakest link”*

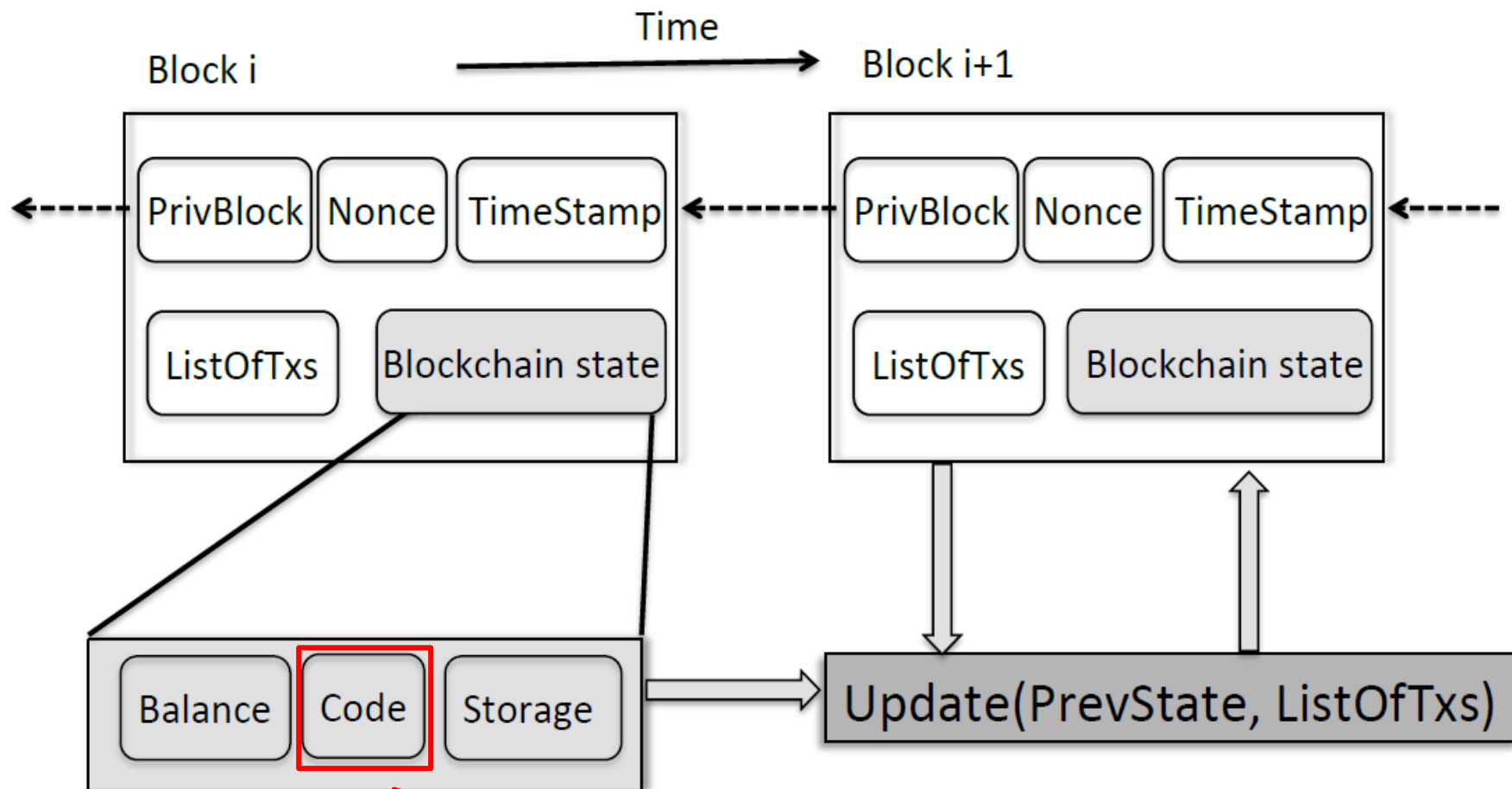


# 1. INTRODUÇÃO

## Ethereum Charts



# 2. BLOCKCHAINS ETHEREUM



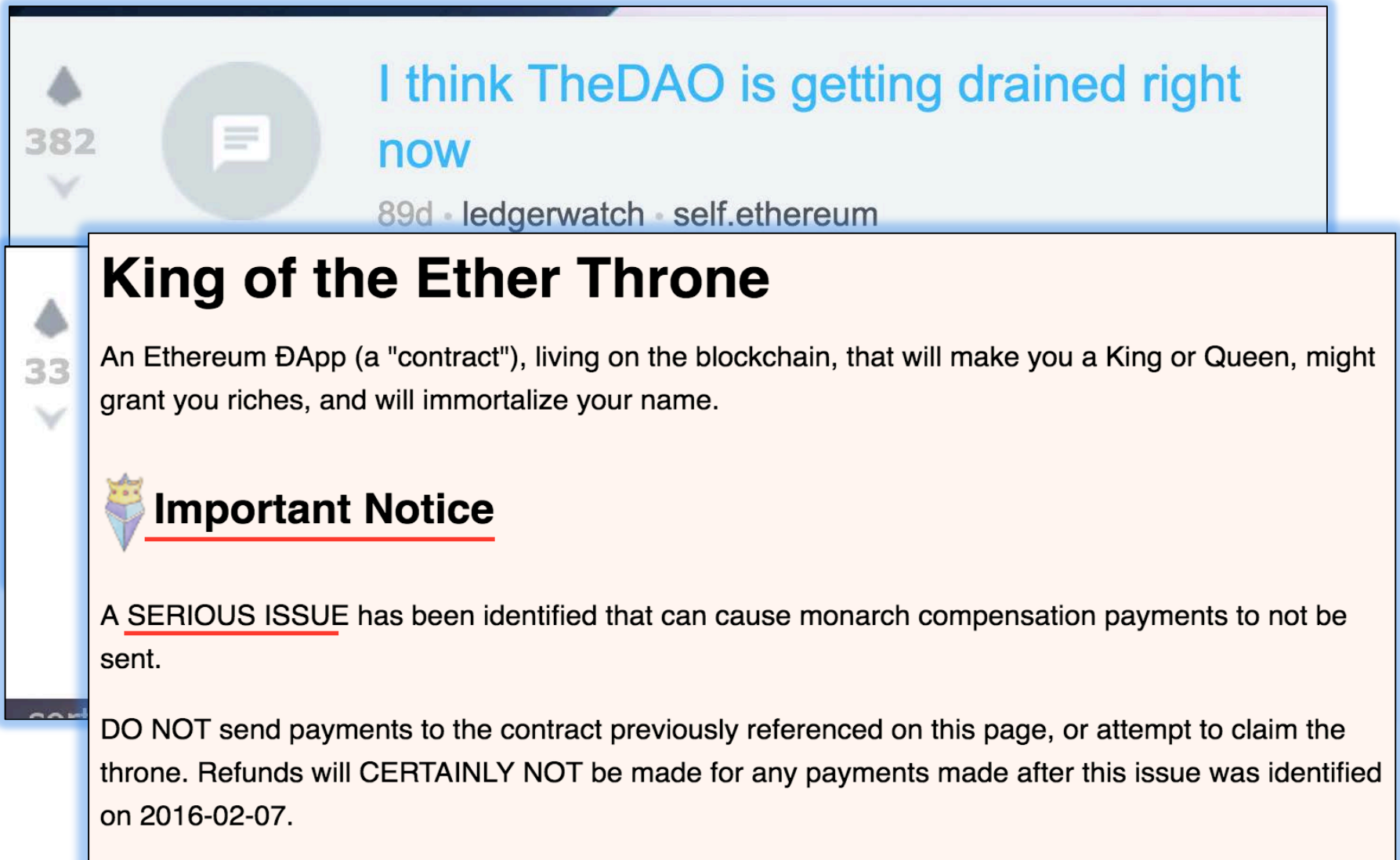
Aqui ficam armazenados os Smart Contracts

# 3. *SMART CONTRACTS (SCs)* E SUAS VULNERABILIDADES

Programar Smart Contracts seguros é bem mais difícil...

- Smart contracts != programas 'normais' porque:
- São auto-executados;
- São impossíveis de corrigir após publicação;
- São escritos em uma nova linguagem (Solidity):
- Solidity != JavaScript
- Solidity != Python

# 3. SMART CONTRACTS (SCs) E SUAS VULNERABILIDADES



The image shows a screenshot of a tweet and a notice from the 'King of the Ether Throne' DApp. The tweet, posted by 'ledgerwatch' on 'self.ethereum' 89 days ago, says 'I think TheDAO is getting drained right now'. Below the tweet is a notice titled 'King of the Ether Throne' with a description of the DApp. The notice includes an 'Important Notice' section with a crown icon, stating that a 'SERIOUS ISSUE' has been identified that could prevent monarch compensation payments. It also includes a warning: 'DO NOT send payments to the contract previously referenced on this page, or attempt to claim the throne. Refunds will CERTAINLY NOT be made for any payments made after this issue was identified on 2016-02-07.'


382

I think TheDAO is getting drained right now

89d · ledgerwatch · self.ethereum

## King of the Ether Throne

An Ethereum DApp (a "contract"), living on the blockchain, that will make you a King or Queen, might grant you riches, and will immortalize your name.

 **Important Notice**

A **SERIOUS ISSUE** has been identified that can cause monarch compensation payments to not be sent.

DO NOT send payments to the contract previously referenced on this page, or attempt to claim the throne. Refunds will **CERTAINLY NOT** be made for any payments made after this issue was identified on 2016-02-07.

# 3. SMART CONTRACTS (SCs) E SUAS VULNERABILIDADES

```
1
2 import "remix_tests.sol";
3 import "./ballot.sol";
4
5 contract test3 {
6
7     Ballot ballotToTest;
8     function beforeAll () public {
9         ballotToTest = new Ballot(2);
10    }
11
12    function checkWinningProposal () public {
13        ballotToTest.vote(1);
14        Assert.equal(ballotToTest.winningProposal(), uint(1), "Pro1 won!");
15    }
16
17    function checkWinninProposalWithReturnValue () public view returns (bool) {
18        return ballotToTest.winningProposal() == 1;
19    }
20 }
21
```

Uma vez carregados para a blockchain, os SmartContracts **não podem mais ser corrigidos!** Por isso a importância de V&V criteriosa **antes da publicação**



# 3. SMART CONTRACTS (SCs) E SUAS VULNERABILIDADES


Cause of Vulnerability	Status
Call to the unknown	Open
Gasless send	Open
Exception disorders	Addressed by Oyente
Type casts	Open
Reentrancy	Addressed by Oyente
Keeping secrets	Open
Immutable bugs	Open
Ether lost in transfer	Open
Stack size limit	Addressed by Oyente
Unpredictable state	Addressed by Oyente
Generating randomness	Open
Integer Underflow & Overflow	Addressed by Oyente
Parity Multisig Bug	Addressed by Oyente
Time constraints	Addressed by Oyente

**Vulnerabilidades que serão endereçadas pela nova versão do Oyente, denominada Oyente-NG (New Generation)**

# 4. DETECÇÃO AUTOMÁTICA DE VULNERABILIDADES EM SMART CONTRACTS

## An Illustrative Example

```
int foo(int v) {  
    return 2*v;  
}  
  
void test_me(int x, int y) {  
    int z = foo(y);  
    if (z == x)  
        if (x > y+10)  
            ERROR;  
}
```



Concrete Execution

concrete state

$x = 2$   
 $y = 1$   
 $z = 2$

Symbolic Execution

symbolic state

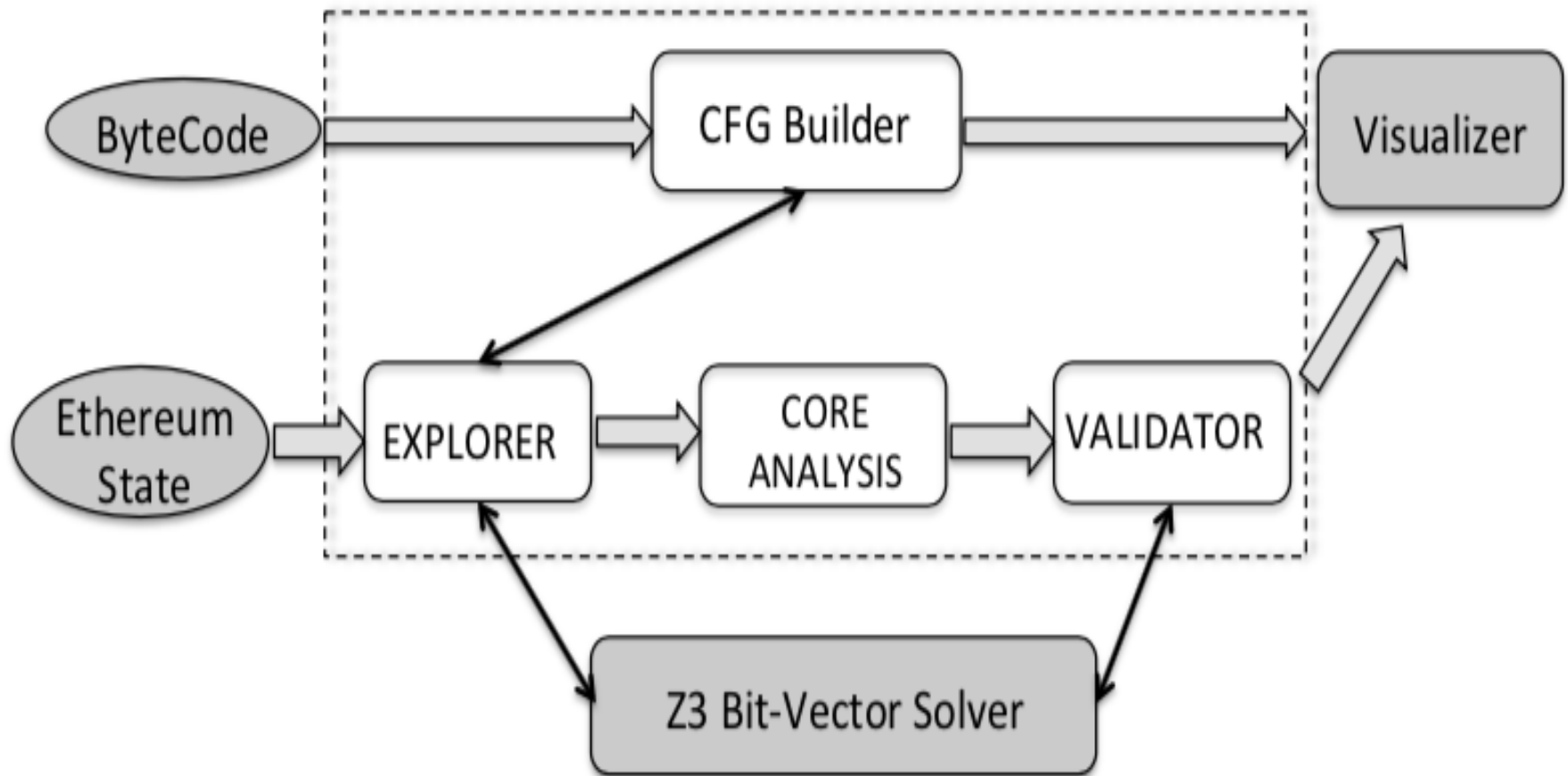
$x = x_\theta$   
 $y = y_\theta$   
 $z = 2*y_\theta$

path condition

$2*y_\theta == x_\theta$   
 $x_\theta \leq y_\theta + 10$

Solve:  $(2*y_\theta == x_\theta)$  and  $(x_\theta > y_\theta + 10)$

# 4. DETECÇÃO AUTOMÁTICA DE VULNERABILIDADES EM SMART CONTRACTS



# 4. DETECÇÃO AUTOMÁTICA DE VULNERABILIDADES EM SMART CONTRACTS

```
oyente-ng -- -bash -- 104x39
Rafaelshigemura$ python oyente.py -s contract_code/nacs/NacsToken.sol
WARNING:root:You are using an untested version of z3. 4.5.1 is the officially tested version
WARNING:root:You are using evm version 1.8.13. The supported version is 1.7.3
WARNING:root:You are using solc version 0.4.24, The latest supported version is 0.4.19
INFO:root:contract contract_code/nacs/NacsToken.sol:NacsToken:
INFO:symExec: ===== Results =====
INFO:symExec: EVM Code Coverage: 99.9%
INFO:symExec: Integer Underflow: False
INFO:symExec: Integer Overflow: True
INFO:symExec: Parity MultiSig Bug 2: False
INFO:symExec: Callstack Depth Attack Vulnerability: False
INFO:symExec: Transaction-Ordering Dependence (TOD): False
INFO:symExec: Timestamp Dependency: False
INFO:symExec: Re-Entrancy Vulnerability: False
INFO:symExec:contract_code/nacs/NacsToken.sol:96:9: Warning: Integer Overflow.
    balanceOf[_to] += _value
Integer Overflow can occur on variables:
    _value
    balanceOf[_to]
    allowance[_from][msg.sender]
    balanceOf[_from]
contract_code/nacs/NacsToken.sol:49:9: Warning: Integer Overflow.
    balanceOf[_to] += _value
Integer Overflow can occur on variables:
    balanceOf[_to]
    _value
    balanceOf[msg.sender]
INFO:symExec: Call to The Unknown: False
INFO:symExec: Ether Lost: False
INFO:symExec: Out-of-Gas Send: True
INFO:symExec: Worst case Gas: 12641
INFO:symExec: Randomness Bug: False
INFO:symExec: Type Cast Vulnerability: False
INFO:symExec: ===== Analysis Completed =====
Rafaelshigemura$
```


## 4. DETECÇÃO AUTOMÁTICA DE VULNERABILIDADES EM SMART CONTRACTS

```
INFO:symExec:contract_code/nacs/NacsToken.sol:96:9: Warning: Integer Overflow.  
    balanceOf[_to] += _value  
Integer Overflow can occur on variables:  
  _value  
  balanceOf[_to]  
  allowance[_from][msg.sender]  
  balanceOf[_from]  
-----  
contract_code/nacs/NacsToken.sol:49:9: Warning: Integer Overflow.  
    balanceOf[_to] += _value  
Integer Overflow can occur on variables:  
  balanceOf[_to]  
  _value  
  balanceOf[msg.sender]
```

```
INFO:symExec:      Out-of-Gas Send:      True  
INFO:symExec:      Worst case Gas:      12641
```


# 4. DETECÇÃO AUTOMÁTICA DE VULNERABILIDADES EM SMART CONTRACTS

```
oyente-ng — -bash — 70x17
INFO:root:contract contract_code/wibx/BCHHandled.sol:BCHHandled:
INFO:symExec: ===== Results =====
INFO:symExec:   EVM Code Coverage:                99.7%
INFO:symExec:   Integer Underflow:                 False
INFO:symExec:   Integer Overflow:                  False
INFO:symExec:   Parity Multisig Bug 2:             False
INFO:symExec:   Callstack Depth Attack Vulnerability:
INFO:symExec:   Transaction-Ordering Dependence (TOD):
INFO:symExec:   Timestamp Dependency:
INFO:symExec:   Re-Entrancy Vulnerability:
INFO:symExec:   Call to The Unknown:
INFO:symExec:   Ether Lost:                        False
INFO:symExec:   Out-of-Gas Send:                   True
INFO:symExec:       Worst case Gas:                 5379
INFO:symExec:   Randomness Bug:                    False
INFO:symExec:   Type Cast Vulnerability:           False
INFO:symExec: ===== Analysis Completed =====
```




# 4. DETECÇÃO AUTOMÁTICA DE VULNERABILIDADES EM SMART CONTRACTS

```
oyente-ng — -bash — 70x17
INFO:root:contract contract_code/wibx/ERC20.sol:ERC20:
INFO:symExec: ===== Results =====
INFO:symExec: EVM Code Coverage: 99.9%
INFO:symExec: Integer Underflow: False
INFO:symExec: Integer Overflow: False
INFO:symExec: Parity Multisig Bug 2: False
INFO:symExec: Callstack Depth Attack Vulnerability:
INFO:symExec: Transaction-Ordering Dependence (TOD):
INFO:symExec: Timestamp Dependency:
INFO:symExec: Re-Entrancy Vulnerability:
INFO:symExec: Call to The Unknown:
INFO:symExec: Ether Lost:
INFO:symExec: Out-of-Gas Send: True
INFO:symExec: Worst case Gas: 6124
INFO:symExec: Randomness Bug: False
INFO:symExec: Type Cast Vulnerability: False
INFO:symExec: ===== Analysis Completed =====
```

A yellow warning triangle icon with a white exclamation mark inside, positioned on the right side of the terminal window, overlapping the text output.

# 4. DETECÇÃO AUTOMÁTICA DE VULNERABILIDADES EM SMART CONTRACTS


```
oyente-ng — -bash — 70x17
INFO:root:contract contract_code/wibx/SafeMath.sol:SafeMath:
INFO:symExec:      ===== Results =====
INFO:symExec:      EVM Code Coverage:                100.0%
INFO:symExec:      Integer Underflow:                   False
INFO:symExec:      Integer Overflow:                     False
INFO:symExec:      Parity Multisig Bug 2:                 False
INFO:symExec:      Callstack Depth Attack Vulnerability: False
INFO:symExec:      Transaction-Ordering Dependence (TOD): False
INFO:symExec:      Timestamp Dependency:                 False
INFO:symExec:      Re-Entrancy Vulnerability:            False
INFO:symExec:      Call to The Unknown:                  False
INFO:symExec:      Ether Lost:                           False
INFO:symExec:      Out-of-Gas Send:                       False
INFO:symExec:      Randomness Bug:                       False
INFO:symExec:      Type Cast Vulnerability:              False
INFO:symExec:      ===== Analysis Completed =====
```






# 4. DETECÇÃO AUTOMÁTICA DE VULNERABILIDADES EM SMART CONTRACTS

```
oyente-ng — -bash — 70x17
INFO:root:contract contract_code/wibx/TaxLib.sol:TaxLib:
INFO:symExec:  ===== Results =====
INFO:symExec:  EVM Code Coverage: 100.0%
INFO:symExec:  Integer Underflow: False
INFO:symExec:  Integer Overflow: False
INFO:symExec:  Parity Multisig Bug 2: False
INFO:symExec:  Callstack Depth Attack Vulnerability: False
INFO:symExec:  Transaction-Ordering Dependence (TOD): False
INFO:symExec:  Timestamp Dependency: False
INFO:symExec:  Re-Entrancy Vulnerability: False
INFO:symExec:  Call to The Unknown: False
INFO:symExec:  Ether Lost: False
INFO:symExec:  Out-of-Gas Send: False
INFO:symExec:  Randomness Bug: False
INFO:symExec:  Type Cast Vulnerability: False
INFO:symExec:  ===== Analysis Completed =====
```



# 4. DETECÇÃO AUTOMÁTICA DE VULNERABILIDADES EM SMART CONTRACTS

```
oyente-ng — -bash — 80x22
INFO:root:contract contract_code/wibx/WibxToken.sol:WibxToken:
INFO:symExec: ===== Results =====
INFO:symExec:      EVM Code Coverage:                82.2%
INFO:symExec:      Integer Underflow:                  True
INFO:symExec:      Integer Overflow:                   False
INFO:symExec:      Parity Multisig Bug 2:               False
INFO:symExec:      Callstack Depth Attack Vulnerability: False
INFO:symExec:      Transaction-Ordering Dependence (TOD): False
INFO:symExec:      Timestamp Dependency:                False
INFO:symExec:      Re-Entrancy Vulnerability:           False
INFO:symExec:contract_code/wibx/WibxToken.sol:18:14: Warning: Integer Underflow.
      uint8 private constan
contract_code/wibx/WibxToken.sol:23:55: Warning: Integer Underflow.
      constructor(address bchAddress) public ERC20Detailed("", "", 18)
INFO:symExec:      Call to The Unknown:                False
INFO:symExec:      Ether Lost:                          False
INFO:symExec:      Out-of-Gas Send:                      True
INFO:symExec:      Worst case Gas:                      6278
INFO:symExec:      Randomness Bug:                      False
INFO:symExec:      Type Cast Vulnerability:             False
INFO:symExec: ===== Analysis Completed =====
```



# 4. CONCLUSÕES

As principais contribuições deste trabalho foram:

1. A proposta de uma taxonomia de vulnerabilidades envolvendo Smart Contracts para Ethereum;
2. A verificação de aplicabilidade da técnica de Execução Simbólica para detecção de vulnerabilidades em Smart Contracts para Ethereum; e
3. A implementação da ferramenta Oyente-NG, que permitiu a detecção de 5 vulnerabilidades adicionais em complemento às 7 existentes anteriormente implementadas na versão original;

# 4. TRABALHOS FUTUROS

1. Aplicar a ferramenta Oyente-NG sobre contratos destinados a outras aplicações, para fins de benchmarking;
2. A implementação de detecção automática de novas vulnerabilidades logo que são descobertas, de forma semelhante às atualizações de produtos antivírus; e
3. Por fim, um estudo de aplicabilidade do conceito proposto para outras linguagens utilizadas para o desenvolvimento de *smart contracts*, por exemplo, *typescript*.

# AGRADECIMENTOS

Ao **Instituto Tecnológico de Aeronáutica (ITA)**, pelo suporte ao longo do desenvolvimento desta pesquisa,

À **Fundação Casimiro Montenegro Filho (FCMF)**, por sua infraestrutura, e

À empresa **Ecosystema Negócios Digitais** por seu apoio financeiro para o desenvolvimento desta e de outras pesquisas, permitindo sua Prova de Conceito em um ambiente real.

# PERGUNTAS?

(rafael@ita.br)