스마트 컨트랙트 안전성 검증

오학주
고려대학교 정보대학 컴퓨터학과

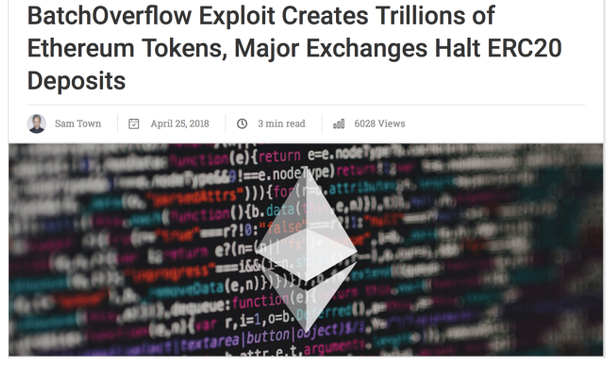
2018.11.27@신한그룹 교류회

연구 분야 소개

- 소프트웨어가 사회 각 영역에서 사용되면서 심각해진 안전성 문제

금융 거래 소프트웨어 결함 (2012)

자율주행 소프트웨어 결함 (2017)



블록체인 소프트웨어 결함 (2018)

연구 분야 소개

- 소프트웨어가 사회 각 영역에서 사용되면서 심각해진 안전성 문제

Knight Capital Says Trading Glitch Cost It $440 Million

BY NATHANIEL POPPER AUGUST 2, 2012 9:07 AM 356

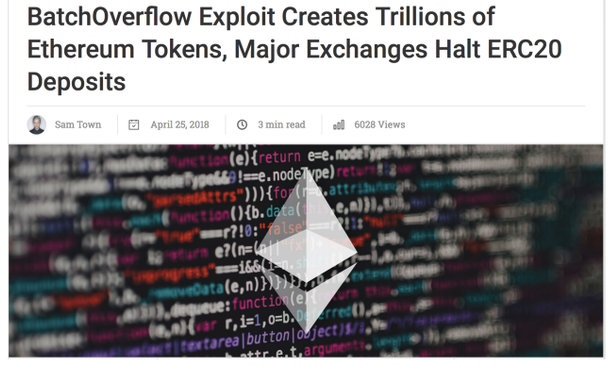
Runaway Trades Spread Turmoil Across Wall St.

금융 거래 소프트웨어 결함 (2012)

Tesla in fatal California crash was on Autopilot

31 March 2018 Share

자율주행 소프트웨어 결함 (2017)

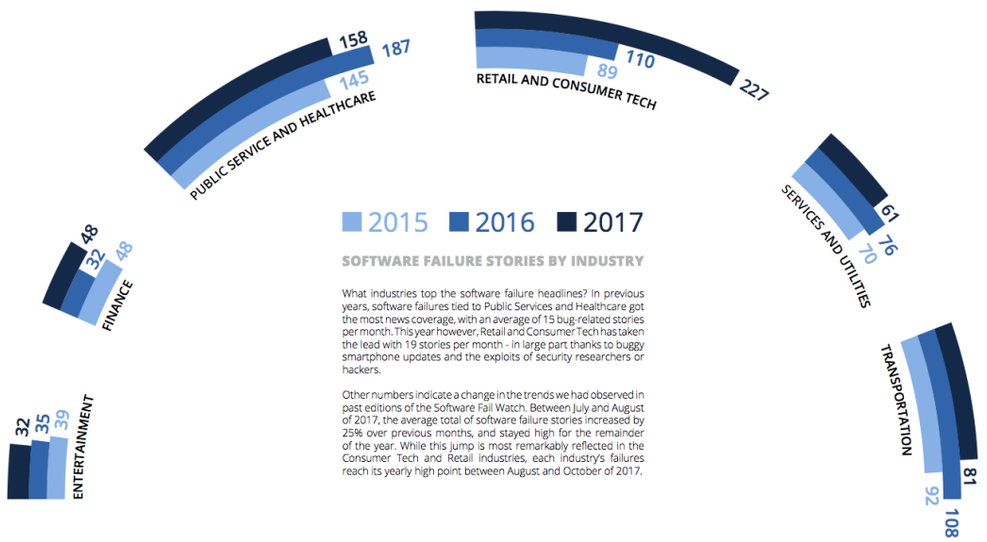


BatchOverflow Exploit Creates Trillions of Ethereum Tokens, Major Exchanges Halt ERC20 Deposits

Sam Town April 25, 2018 3 min read 6028 Views

블록체인 소프트웨어 결함 (2018)

- 사회 모든 영역의 문제 (금융, 헬스케어, 가전, 공공, 교통, ...)



2000조원

Software fail watch (5th ed) 2017

연구 분야 소개

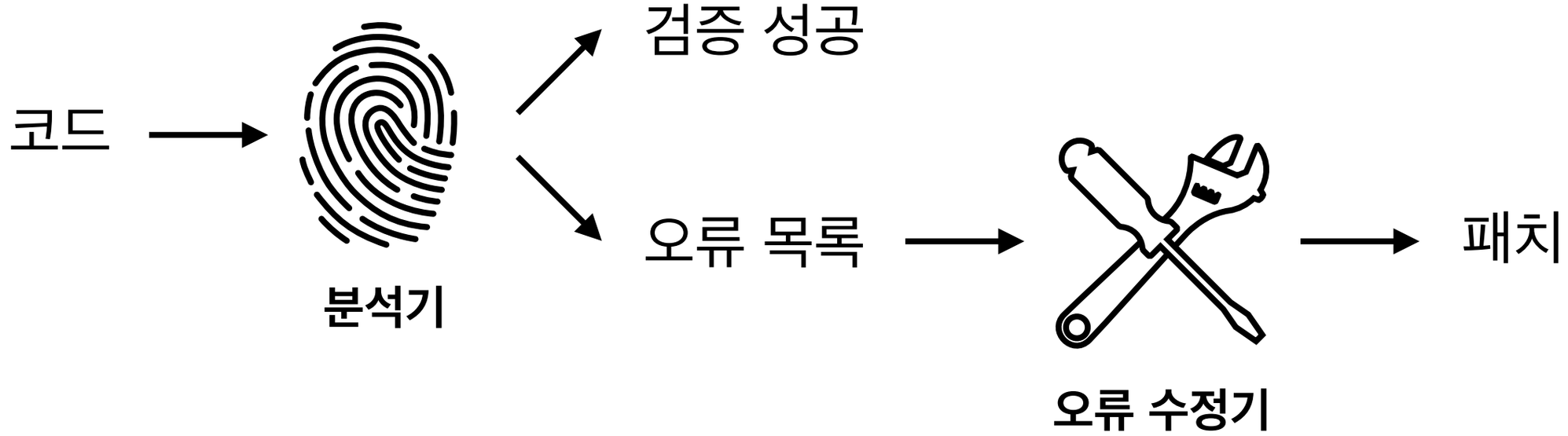
- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성**

연구 분야 소개

- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성**

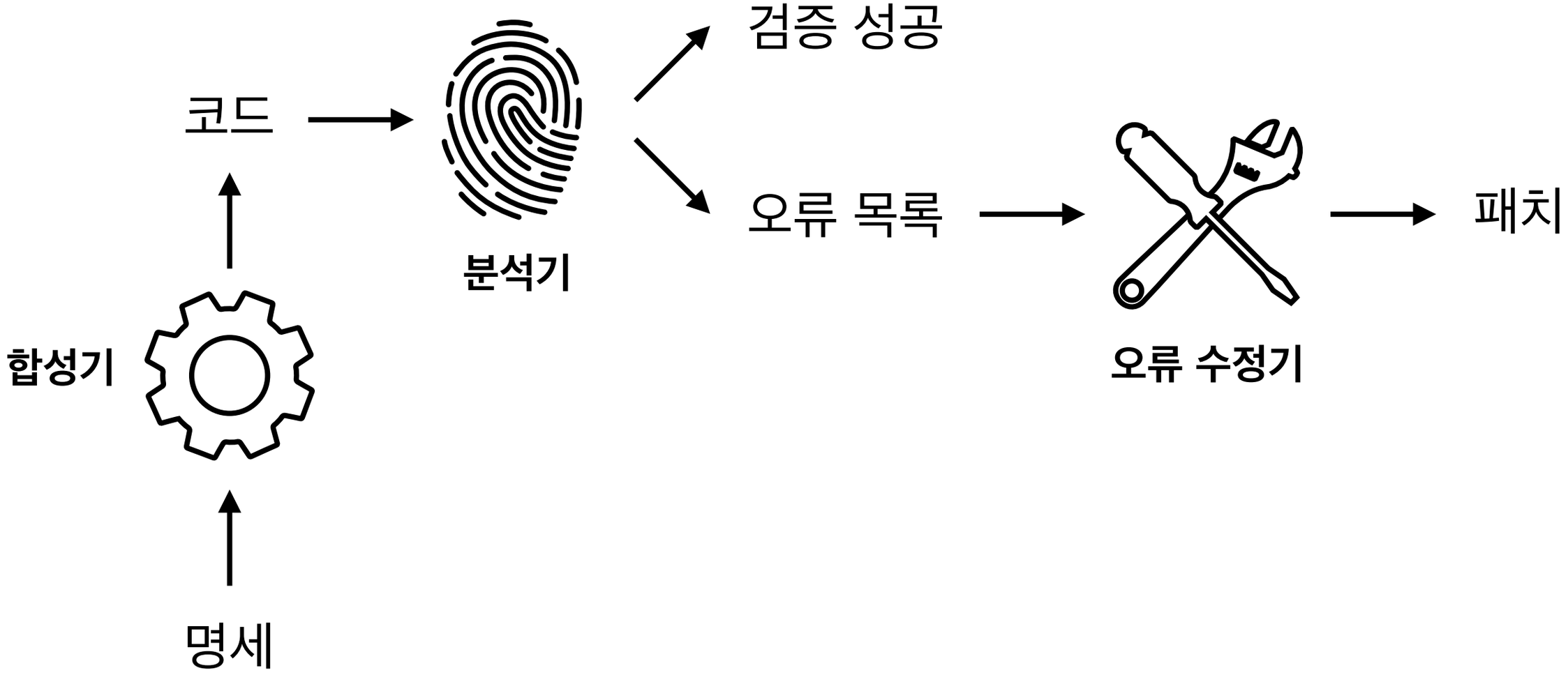
연구 분야 소개

- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성**



연구 분야 소개

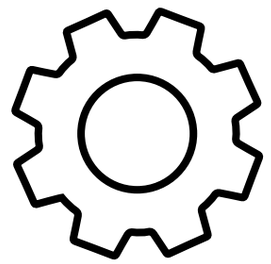
- Q) 어떻게 안전한 소프트웨어를 손쉽게 만들것인가?
- A) 소프트웨어 자동 **분석**, **패치**, **합성**



연구 분야 소개

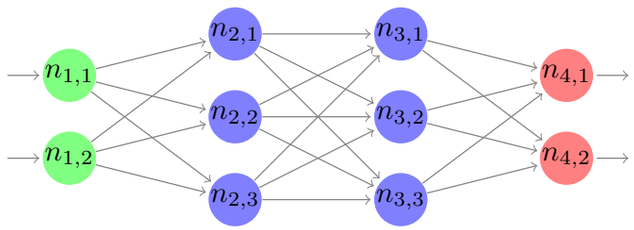
- 연구방향 1: 소프트웨어 분석, 패치, 합성을 위한 범용 원천 기술





- 연구방향 2: 응용 분야별로 특화한 실용 기술

응용/시스템 소프트웨어



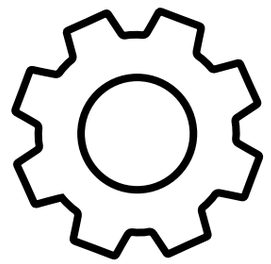
인공지능 소프트웨어

블록체인 소프트웨어

연구 분야 소개

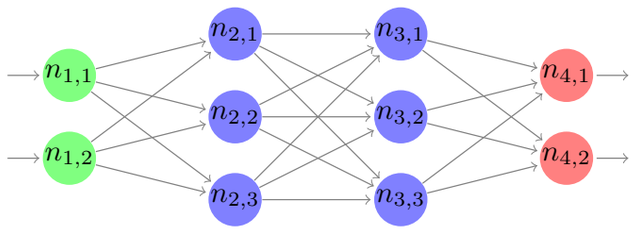
- 연구방향 1: 소프트웨어 분석, 패치, 합성을 위한 범용 원천 기술





- 연구방향 2: 응용 분야별로 특화한 실용 기술

응용/시스템 소프트웨어



인공지능 소프트웨어

블록체인 소프트웨어

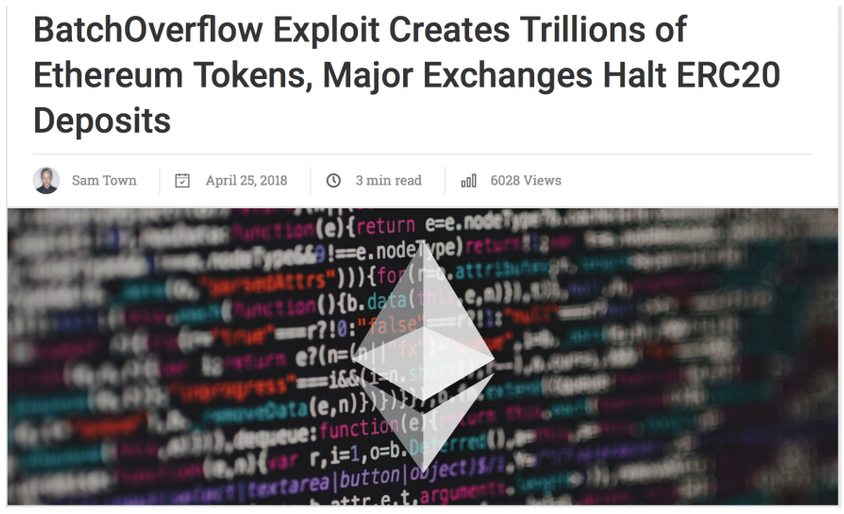
Today

스마트 컨트랙트

- 프로그래밍 언어로 작성된 계약서 (블록체인에서 실행)
  - 금융 거래, 부동산 거래, 보험, 공증 등
- 스마트 컨트랙트의 오류는 심각한 금전적 피해를 초래

A $50 MILLION HACK JUST SHOWED THAT THE DAO WAS ALL TOO HUMAN

The DAO (2016)
750억원



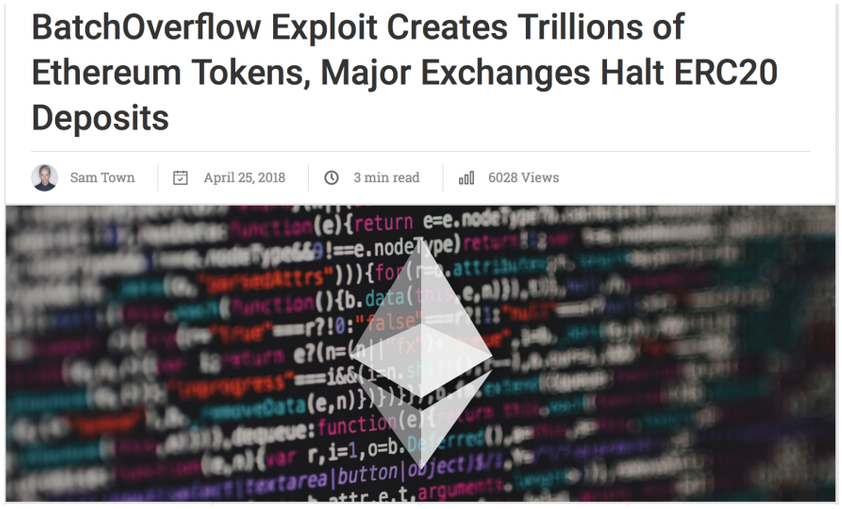
SmartMesh (2018)
천문학적 금액 인출 시도

VeriSmart

- **Veri**fication of **Smart** contract
  - 스마트 컨트랙트를 위한 분석, 패치, 합성 프레임워크
- 경쟁력: 지난 10여년 축적한 소프트웨어 분석 기술 위에 구축

성능

- **모든** 대상 오류를 **정확**하고 **빠르게** 탐지
- 기존 도구 보다 월등한 성능

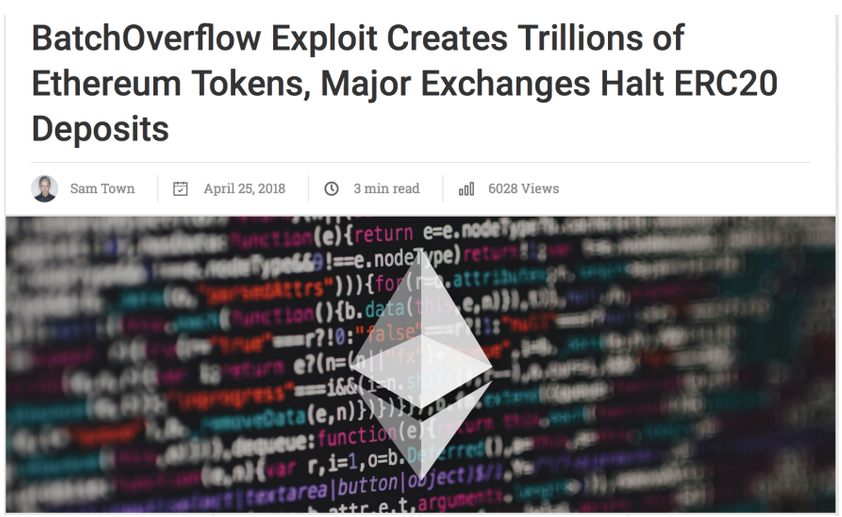


SmartMesh (2018)

```
// Abstract contract for the full ERC 20 Token standard
// https://github.com/ethereum/EIPs/issues/20
pragma solidity ^0.4.15;

contract Token {
    /* This is a slight change to the ERC20 base standard.*/
    /// total amount of tokens
    uint256 public totalSupply;

    /// @param _owner The address from which the balance will be retrieved
    /// @return The balance
    function balanceOf(address _owner) public constant returns (uint256 balance);

    ///@notice send `_value` token to `_to` from `msg.sender`
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transfer(address _to, uint256 _value) public returns (bool success);

    /// @notice send `_value` token to `_to` from `_from` on the condition it is approved by `_from`
    /// @param _from The address of the sender
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);

    /// @notice `msg.sender` approves `_spender` to spend `_value` tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @param _value The amount of tokens to be approved for transfer
    /// @return Whether the approval was successful or not
    function approve(address _spender, uint256 _value) public returns (bool success);

    /// @param _owner The address of the account owning tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @return Amount of remaining tokens allowed to spent
    function allowance(address _owner, address _spender) public constant returns (uint256 remaining);

    event Transfer(address indexed _from, address indexed _to, uint256 _value);
    event Approval(address indexed _owner, address indexed _spender, uint256 _value);
}

contract Owned {

    /// `owner` is the only address that can call a function with this
    /// modifier
    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    address public owner;

    /// @notice The Constructor assigns the message sender to be `owner`
    function Owned() public {
        owner = msg.sender;
    }

    address newOwner=0x0;

    event OwnerUpdate(address _prevOwner, address _newOwner);

    ///changes the owner
    function changeOwner(address _newOwner) public onlyOwner {
        require(_newOwner != owner);
        newOwner = _newOwner;
    }

    /// accept the ownership
    function acceptOwnership() public{
        require(msg.sender == newOwner);
        OwnerUpdate(owner, newOwner);
        owner = newOwner;
        newOwner = 0x0;
    }
}

contract Controlled is Owned{

    function Controlled() public {
        setExclude(msg.sender);
    }

    // Flag that determines if the token is transferable or not.
    bool public transferEnabled = false;

    // flag that makes locked address effect
    bool lockFlag=true;
    mapping(address => bool) locked;
    mapping(address => bool) exclude;

    function enableTransfer(bool _enable) public onlyOwner{
        transferEnabled=_enable;
    }

    function disableLock(bool _enable) public onlyOwner returns (bool success){
        lockFlag=_enable;
        return true;
    }

    function addLock(address _addr) public onlyOwner returns (bool success){
        require(_addr!=msg.sender);
        locked[_addr]=true;
        return true;
    }

    function setExclude(address _addr) public onlyOwner returns (bool success){
        exclude[_addr]=true;
        return true;
    }

    function removeLock(address _addr) public onlyOwner returns (bool success){
        locked[_addr]=false;
        return true;
    }

    modifier transferAllowed(address _addr) {
        if (!exclude[_addr]) {
            assert(transferEnabled);
            if(lockFlag){
                assert(!locked[_addr]);
            }
        }
        _;
    }

}

contract StandardToken is Token,Controlled {

    function transfer(address _to, uint256 _value) public transferAllowed(msg.sender) returns (bool success) {
        //Default assumes totalSupply can't be over max (2^256 - 1).
        //If your token leaves out totalSupply and can issue more tokens as time goes on, you need to check if it doesn'
        //Replace the if with this one instead.
        if (balances[msg.sender] >= _value && balances[_to] + _value > balances[_to]) {
            balances[msg.sender] -= _value;
            balances[_to] += _value;
            Transfer(msg.sender, _to, _value);
            return true;
        } else { return false; }
    }

    function transferFrom(address _from, address _to, uint256 _value) public transferAllowed(_from) returns (bool success
        //same as above. Replace this line with the following if you want to protect against wrapping uints.
        if (balances[_from] >= _value && allowed[_from][msg.sender] >= _value && balances[_to] + _value > balances[_to])
            balances[_from] -= _value;
            balances[_to] += _value;
            allowed[_from][msg.sender] -= _value;
            Transfer(_from, _to, _value);
            return true;
        } else { return false; }
    }

    function balanceOf(address _owner) public constant returns (uint256 balance) {
        return balances[_owner];
    }

    function approve(address _spender, uint256 _value) public returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }

    function allowance(address _owner, address _spender) public constant returns (uint256 remaining) {
      return allowed[_owner][_spender];
    }

    mapping (address => uint256) balances;
    mapping (address => mapping (address => uint256)) allowed;
}

contract SMT is StandardToken {

    function () public {
        revert();
    }

                                                    ///fancy name
    string public name = "SmartMesh Token";
    uint8 public decimals = 18;                     //How many decimals to show. ie. There could 1000 base units with 3 decim
    string public symbol = 'SMT';                   //An identifier
    string public version = 'v0.1';       //SMT 0.1 standard. Just an arbitrary versioning scheme.
    uint256 public allocateEndTime;

    // The nonce for avoid transfer replay attacks
    mapping(address => uint256) nonces;

    function SMT() public {
        allocateEndTime = now + 1 days;
    }

    /*
     * Proxy transfer SmartMesh token. When some users of the ethereum account has no ether,
     * he or she can authorize the agent for broadcast transactions, and agents may charge agency fees
     * @param _from
     * @param _to
     * @param _value
     * @param feeSmt
     * @param _v
     * @param _r
     * @param _s
     */
    function transferProxy(address _from, address _to, uint256 _value, uint256 _feeSmt,
        uint8 _v,bytes32 _r, bytes32 _s) public transferAllowed(_from) returns (bool){

        if(balances[_from] < _feeSmt + _value) revert();

        uint256 nonce = nonces[_from];
        bytes32 h = keccak256(_from,_to,_value,_feeSmt,nonce);
        if(_from != ecrecover(h,_v,_r,_s)) revert();

        if(balances[_to] + _value < balances[_to]
            || balances[msg.sender] + _feeSmt < balances[msg.sender]) revert();
        balances[_to] += _value;
        Transfer(_from, _to, _value);

        balances[msg.sender] += _feeSmt;
        Transfer(_from, msg.sender, _feeSmt);

        balances[_from] -= _value + _feeSmt;
        nonces[_from] = nonce + 1;
        return true;
    }

    /*
     * Proxy approve that some one can authorize the agent for broadcast transaction
     * which call approve method, and agents may charge agency fees
     * @param _from The address which should transfer SMT to others
     * @param _spender The spender who allowed by _from
     * @param _value The value that should be transfered.
     * @param _v
     * @param _r
     * @param _s
     */
    function approveProxy(address _from, address _spender, uint256 _value,
        uint8 _v,bytes32 _r, bytes32 _s) public returns (bool success) {

        uint256 nonce = nonces[_from];
        bytes32 hash = keccak256(_from,_spender,_value,nonce);
        if(_from != ecrecover(hash,_v,_r,_s)) revert();
        allowed[_from][_spender] = _value;
        Approval(_from, _spender, _value);
        nonces[_from] = nonce + 1;
        return true;
    }


    /*
     * Get the nonce
     * @param _addr
     */
    function getNonce(address _addr) public constant returns (uint256){
        return nonces[_addr];
    }

    /* Approves and then calls the receiving contract */
    function approveAndCall(address _spender, uint256 _value, bytes _extraData) public returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);

        //call the receiveApproval function on the contract you want to be notified. This crafts the function signature
        //receiveApproval(address _from, uint256 _value, address _tokenContract, bytes _extraData)
        //it is assumed that when does this that the call *should* succeed, otherwise one would use vanilla approve inst
        if(!_spender.call(bytes4(bytes32(keccak256("receiveApproval(address,uint256,address,bytes)"))), msg.sender, _val
        return true;
    }

    /* Approves and then calls the contract code*/
    function approveAndCallcode(address _spender, uint256 _value, bytes _extraData) public returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);

        //Call the contract code
        if(!_spender.call(_extraData)) { revert(); }
        return true;
    }

    // Allocate tokens to the users
    // @param _owners The owners list of the token
    // @param _values The value list of the token
    function allocateTokens(address[] _owners, uint256[] _values) public onlyOwner {

        if(allocateEndTime < now) revert();
        if(_owners.length != _values.length) revert();

        for(uint256 i = 0; i < _owners.length ; i++){
            address to = _owners[i];
            uint256 value = _values[i];
            if(totalSupply + value <= totalSupply || balances[to] + value <= balances[to]) revert();
            totalSupply += value;
            balances[to] += value;
        }
    }
}
```

성능

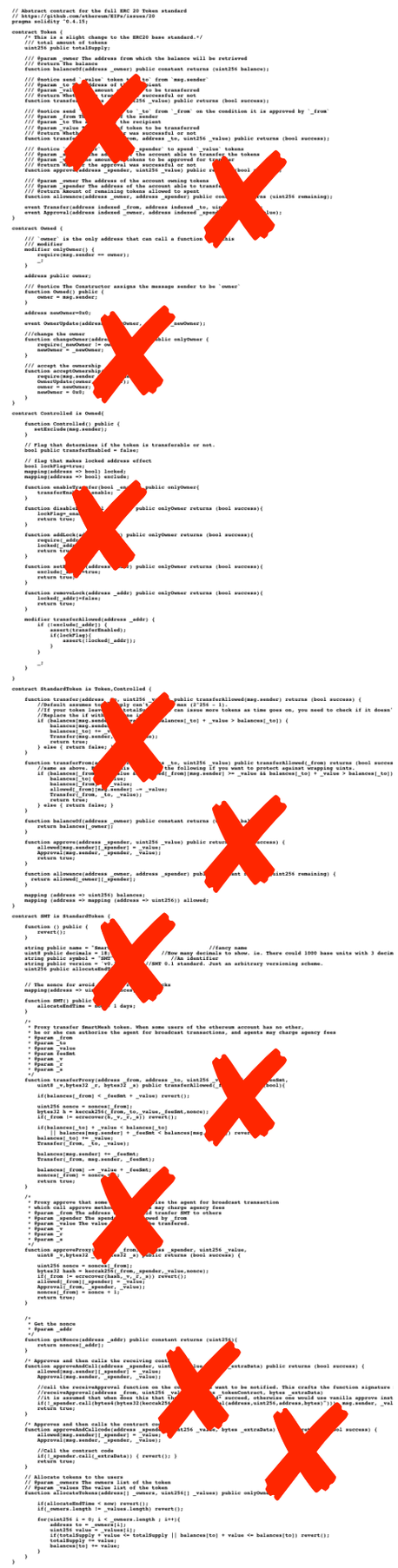
- **모든** 대상 오류를 **정확**하고 **빠르게** 탐지
- 기존 도구 보다 월등한 성능



BatchOverflow Exploit Creates Trillions of Ethereum Tokens, Major Exchanges Halt ERC20 Deposits

Sam Town | April 25, 2018 | 3 min read | 6028 Views

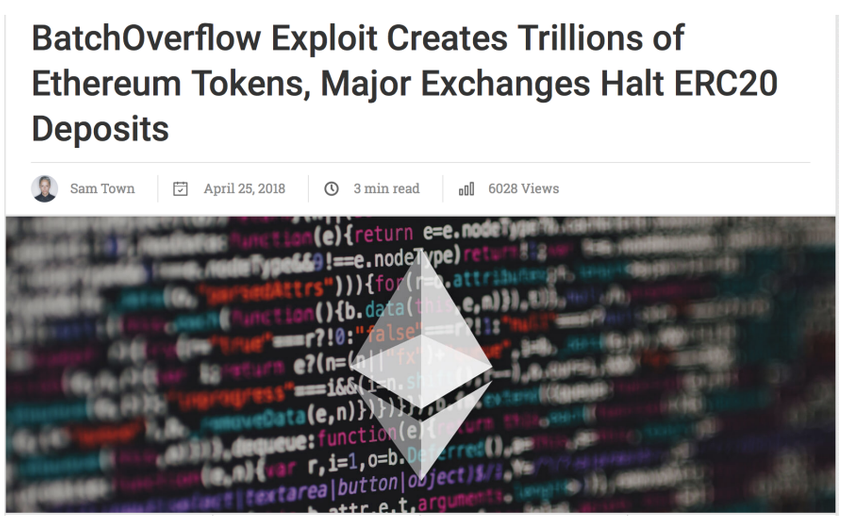
SmartMesh (2018)



VeriSmart

성능

- **모든** 대상 오류를 **정확**하고 **빠르게** 탐지
- 기존 도구 보다 월등한 성능



SmartMesh (2018)

기존 도구
(Mythril, Oyente, etc)

마무리

- **안전한 소프트웨어**를 손쉽게 만들기 위한 기술 연구
  - 소프트웨어 자동 **분석**, **패치**, **합성** 기술에 집중
- 응용 분야:
  - 웹/모바일 소프트웨어, 시스템 소프트웨어, 인공지능 소프트웨어, **블록체인 소프트웨어**, etc

감사합니다!