EVMPatch: Timely and Automated Patching of Ethereum Smart Contracts

USENIX Security 2021

Background

- Smart contract

```
pragma solidity >=0.7.0 <0.9.0;

contract Storage {

    uint256 number;

    function store(uint256 num) public {
        number = num;
    }

    function retrieve() public view returns (uint256){
        return number;
    }
}
```

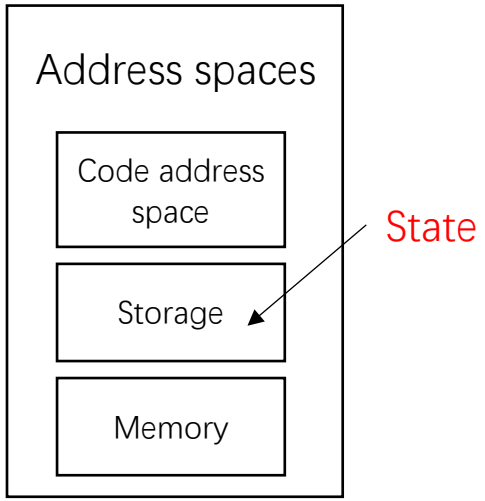
Assembly

```
  PUSH [tag] 8                    function retrieve() public
  JUMP [in]                  function retrieve() public vie...
tag 7                        function retrieve() public vie...
  JUMPDEST                   function retrieve() public vie...
  PUSH 40                    function retrieve() public vie...
  MLOAD                      function retrieve() public vie...
  DUP1                       function retrieve() public vie...
  SWAP2                      function retrieve() public vie...
  SUB                        function retrieve() public vie...
  SWAP1                      function retrieve() public vie...
  RETURN                     function retrieve() public vie...
tag 4                        function store(uint256 num) pu...
  JUMPDEST                   function store(uint256 num) pu...
  PUSH [tag] 9                      function store(uint256 num
  PUSH 4                     function store(uint256 num) pu...
  DUP1                       function store(uint256 num) pu...
  CALLDATASIZE                      function store(uint256 num
  SUB                        function store(uint256 num) pu...
  DUP2                       function store(uint256 num) pu...
  ADD                        function store(uint256 num) pu...
  SWAP1                      function store(uint256 num) pu...
  PUSH [tag] 10                     function store(uint256 num
  SWAP2                      function store(uint256 num) pu...
  SWAP1                      function store(uint256 num) pu...
  PUSH [tag] 11                     function store(uint256 num
  JUMP [in]                  function store(uint256 num) pu...
tag 10                       function store(uint256 num) pu...
  JUMPDEST                   function store(uint256 num) pu...
  PUSH [tag] 12                     function store(uint256 num
  JUMP [in]                  function store(uint256 num) pu...
tag 9                        function store(uint256 num) pu...
  JUMPDEST                   function store(uint256 num) pu...
```

Bytecode

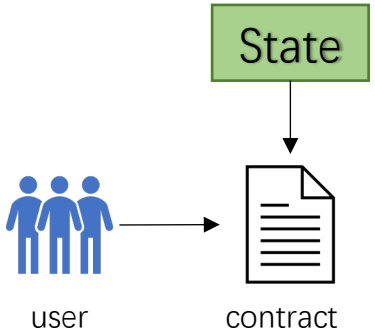
```
608060405234801561001057600080fd5b506
101508061002060000396000f3fe60806040523
48015610010057600080fd5b50600436106100
365760003560e01c80632e64cec11461003b5
780636057361d14610059575b600080fd5b61
0043610075565b604051610050919061006d95
65b60405180910390f35b6100736004803603
81019061006e919061009d565b61007e565b
005b60008054905090565b806000819055505
0565b600081359050610097816101035656b92
915050565b60006020828403121561006100b3576
100b26100fe565b5b6……
```

- EVM：custom instruction format；stack-based；
  resource management mechanism(gas)；
  different address space

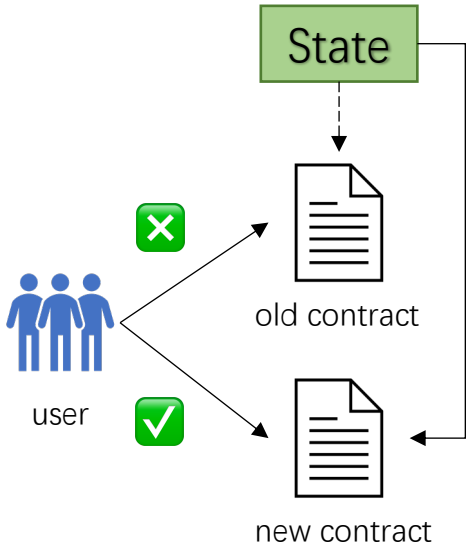


Background

- Contract Upgrade Strategies

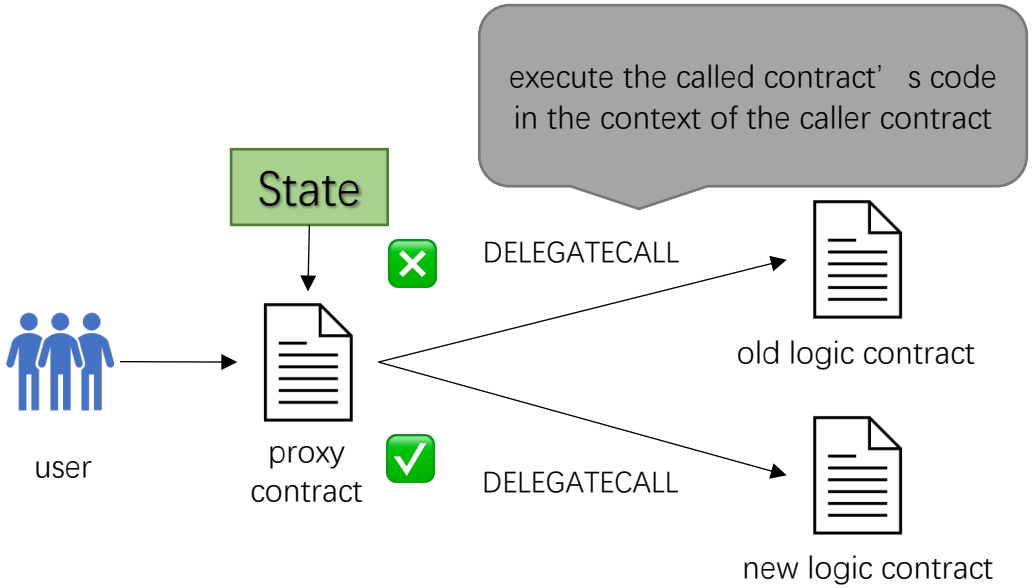


eternal storage pattern



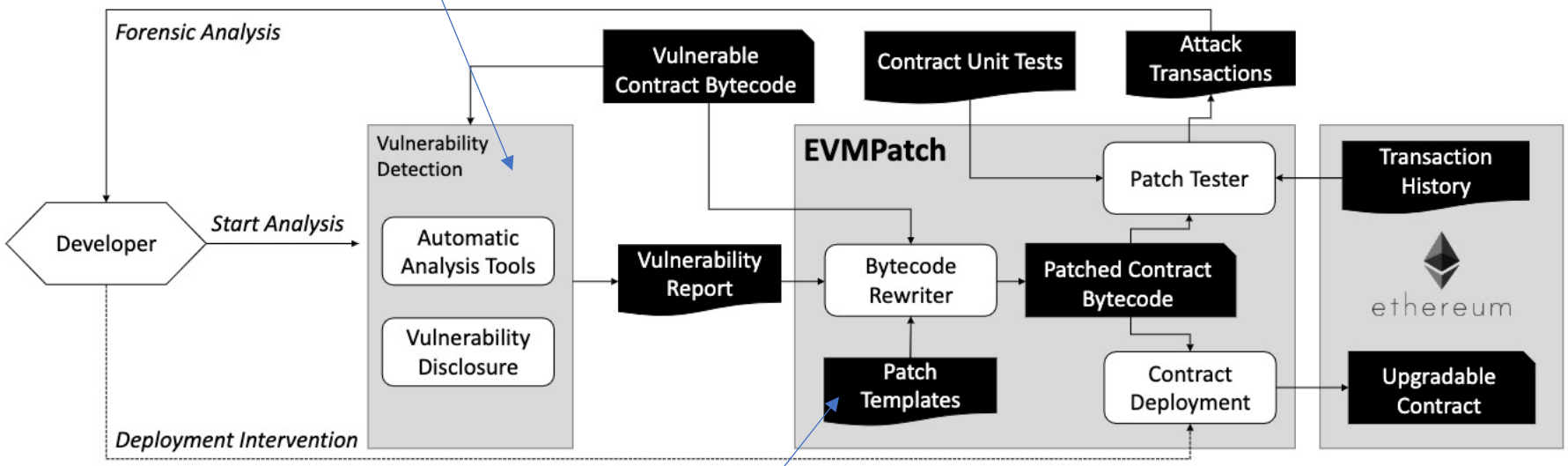
Shortage：
-expensive
-update all contracts that interact with old contracts
-contact all users

delegatecall-proxy pattern



Framework

existing vulnerability detection tools



patch-template for every supported class of vulnerabilities

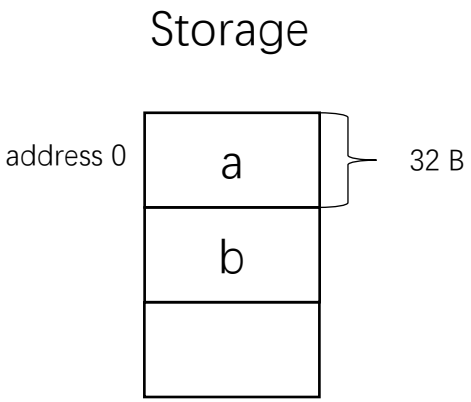
Challenges

- Smart contract automatic rewriting
  - — Correct
  - — High efficiency
- Patch testing
  - — Does not affect the original function
  - — New patch available
- Deployment of patched contracts
  - — Convert to proxy pattern

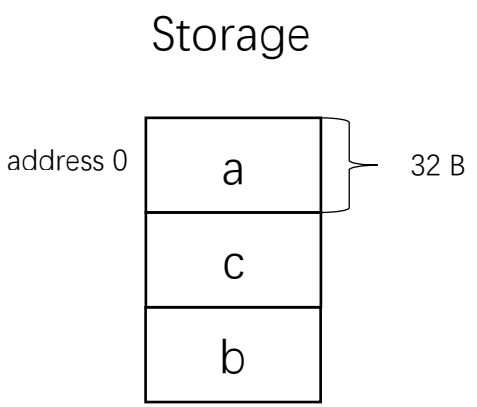
- Contract rewriting-Source VS Bytecode

Source Rewriting	Bytecode Rewriting
Corrupts storage-layout	Preserves storage layout
Checking modifications by human analyst feasible	Human analysis of bytecode changes challenging
Limited tool support for vulnerability analysis	Easy integration of vulnerability analysis tools

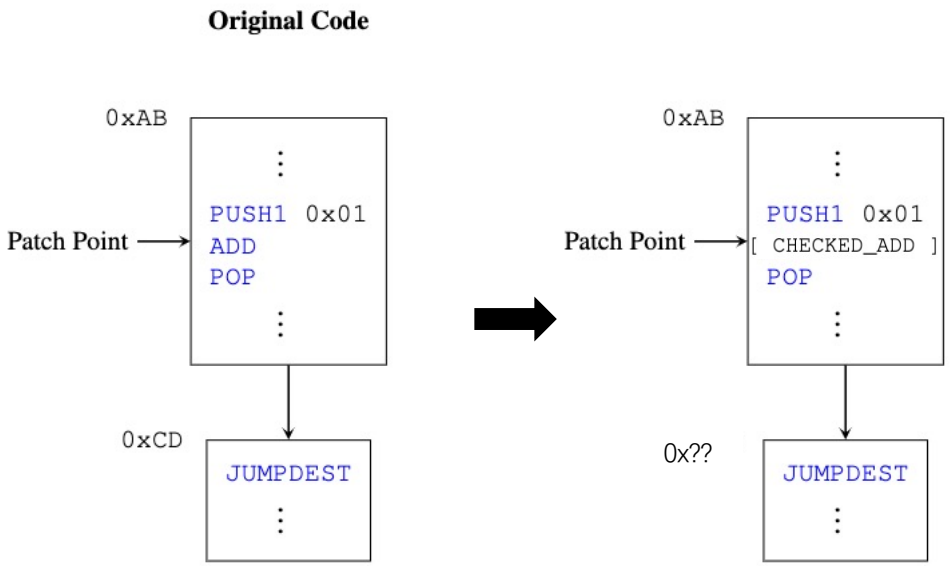
```
int a = 100;
int b = 10;
```

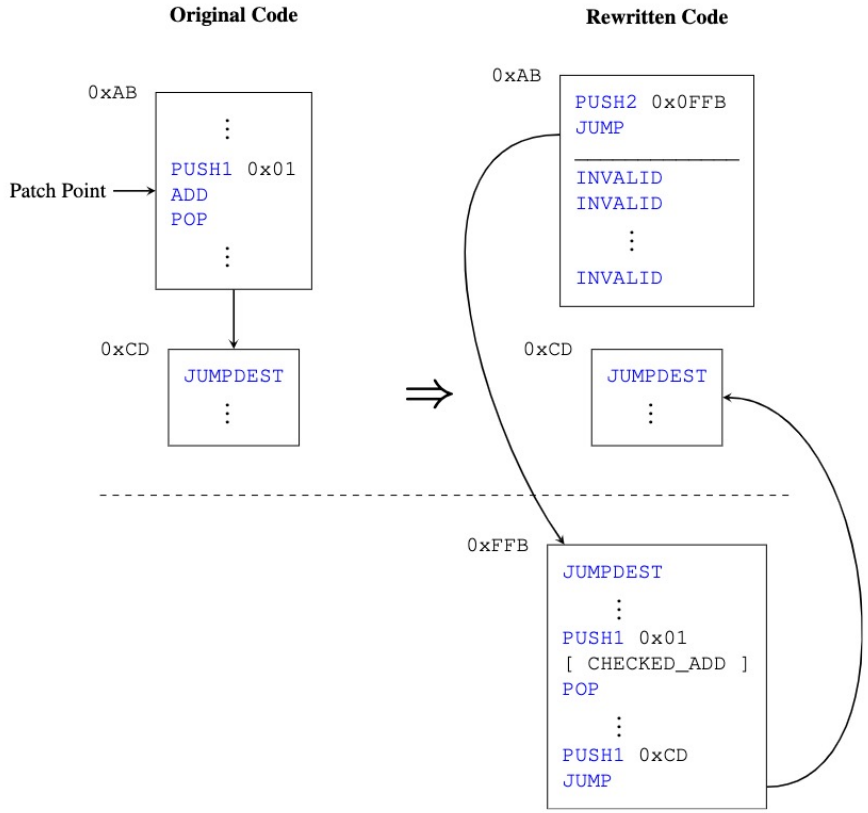



```
int a = 100;
int c = 10;
int b = 200;
```



- Bytecode rewriting-trampoline concept





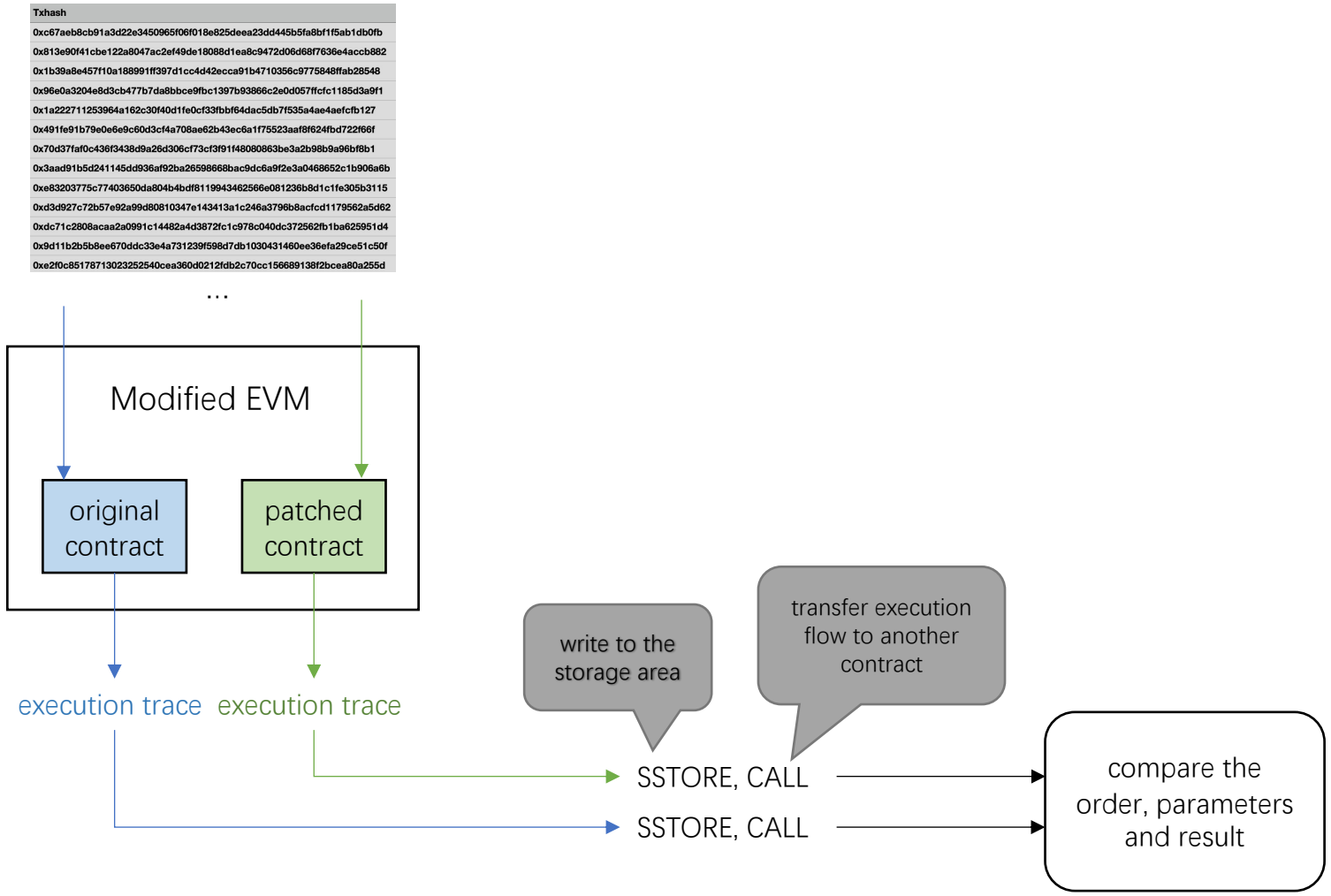
The basic block contains the patch point → Trampoline

exact same size

— Every instruction is represented as a one-byte opcode

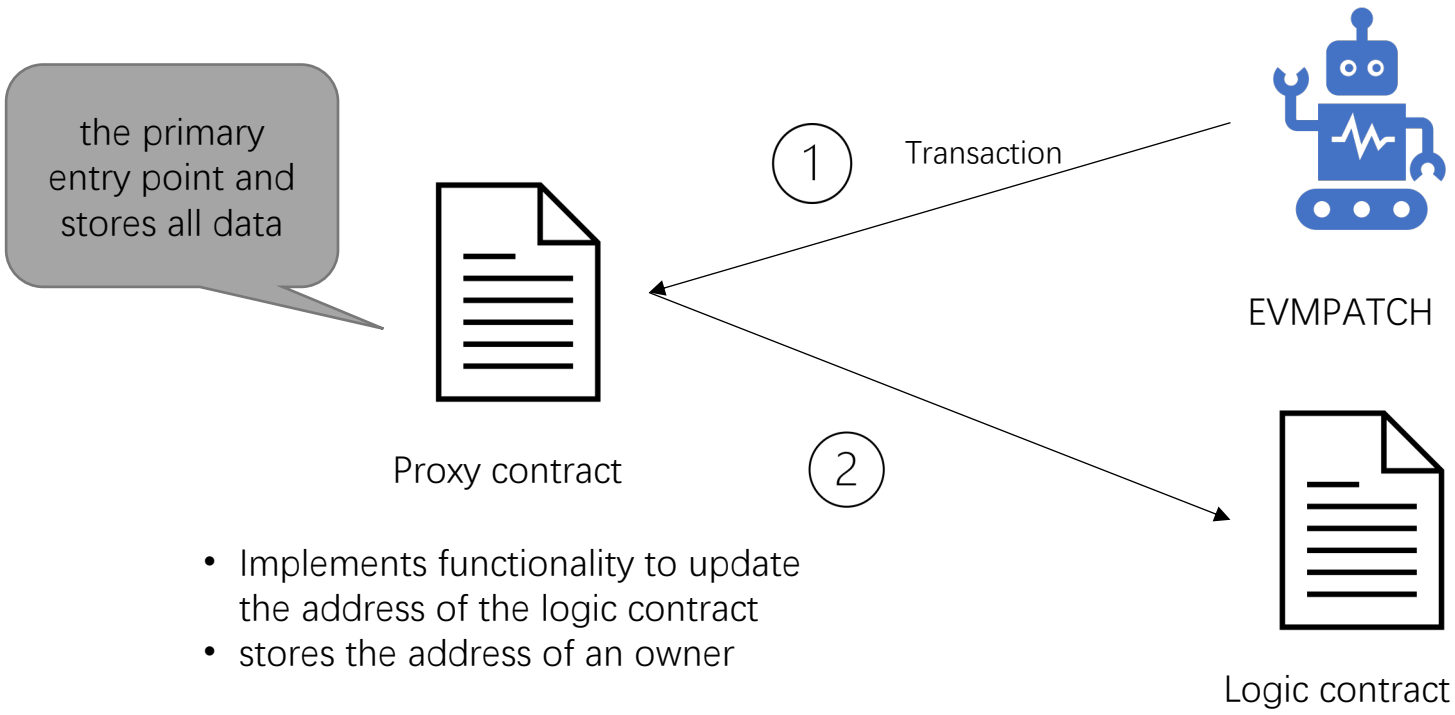
Patch testing

- Does not affect the original function — differential testing approach
- New patch available — unit tests



Deployment of Patched Contracts

— Utilizes a proxy contract that is shipped with EVMPATCH



- Implements functionality to update the address of the logic contract
- stores the address of an owner

43.167 gas≈ $0.01

Evaluation

- Patching Access Control Bugs

Parity Wallet

```
function initWallet(address[] _owners, uint _required, uint _daylimit) {
    initDaylimit(_daylimit);
    initMultiowned(_owners, _required);
}
```

```
function initMultiowned(address[] _owners, uint _required) {
    m_numOwners = _owners.length + 1;
    m_owners[1] = uint(msg.sender);
    m_ownerIndex[uint(msg.sender)] = 1;
    for (uint i = 0; i < _owners.length; ++i) {
        m_owners[2 + i] = uint(_owners[i]);
        m_ownerIndex[uint(_owners[i])] = 2 + i;
    }
    m_required = _required;
}
```

```
function initMultiowned(address[] _owners, uint _required)
                ① internal {
// ...
function initDaylimit(uint _limit) ① internal {
// ...
// throw unless the contract is not yet initialized.
modifier only_uninitialized { if (m_numOwners > 0) throw; _;}

function initWallet(address[] _owners, uint _required,
            uint _daylimit)
            ② only_uninitialized {
// ...
```

Source code of patched Parity Wallet

Overhead of access control patch

Version	Bytes	Size Increase	Gas Increase
Original	8290	0 %	0
Source-Patched	8201	−1.07 %	226
EVMPATCH'ed	8315	0.3 %	235

235 gas≈ $0.0006

```
add_require_patch:
  initWallet:
    - sload(m_numOwner) == 0

delete_public_function_patch:
  - initDayLimit
  - initMultiowned
```

whether the contract is not yet initialized

removes a public function from the public function dispatcher

Customized Patch for Parity Wallet

Evaluation

- Patching Integer Bugs

Integer overflow bug brief

```
function batchTransfer(address[] _receivers, uint256 _value)
                    public whenNotPaused returns (bool) {
  uint cnt = _receivers.length;
// OVERFLOW: 2 * ((INT_MAX / 2) + 1) == 0
  uint256 amount = uint256(cnt) * _value;
  require(cnt > 0 && cnt <= 20);
// BYPASSED CHECK: balances[msg.sender] >= 0
  require(_value > 0 && balances[msg.sender] >= amount);
// RESULT: Transfer of ((INT_MAX / 2) + 1) tokens
```

```
function batchTransfer(address[] _receivers, uint256 _value)
                    public whenNotPaused returns (bool) {
  uint cnt = _receivers.length;
// OVERFLOW: 2 * ((INT_MAX / 2) + 1) == 0
  uint256 amount = uint256(cnt) * _value;      SafeMath.mul(uint256(cnt), _value);
  require(cnt > 0 && cnt <= 20);
// BYPASSED CHECK: balances[msg.sender] >= 0
  require(_value > 0 && balances[msg.sender] >= amount);
// RESULT: Transfer of ((INT_MAX / 2) + 1) tokens
```

Source code of patched Interger overflow contract

Integer patch templates add checks inspired by secure coding rules in the C programming language and the SafeMath Solidity library

Contract	CVE	# Patches	# Transactions		Overhead (gas)		Code Size Increase (B)		Additional Cost RW (US$)	
			Total	Attacks	RW	SM	RW	SM	per TX	per Upgrade
BEC [2]	2018-10299	1	424,229	1	83	164	117 (1.0%)	133 (1.1%)	<0.01	0.01
SMT [36]	2018-10376	1	56,555	1	47	108	191 (0.8%)	97 (0.4%)	<0.01	0.01
UET [43]	2018-10468	55	24,034	12	225	21	1,299 (18.2%)	541 (7.6%)	<0.01	0.071
SCA [37]	2018-10706	1	292	10	47	0	3,811 (17.3%)	361 (1.6%)	<0.01	0.189
HXG [17]	2018-11239	9	1497	5	120	541	997 (28.1%)	519 (14.6%)	<0.01	0.057

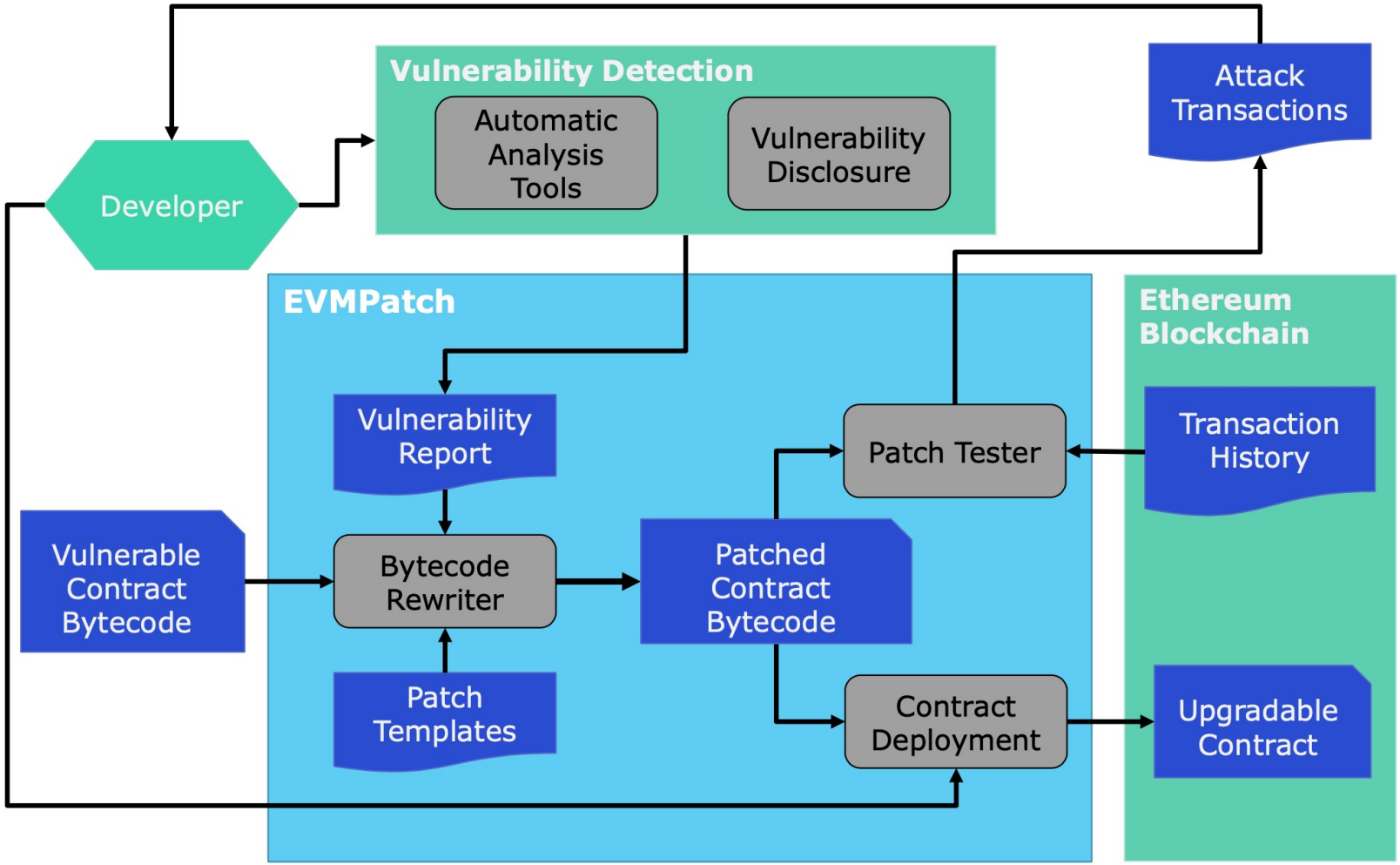
Evaluation

- Developer Study

1. manually patch three contracts vulnerable due to integer overflow bugs given the output of a static analyze.
2. convert a contract to an upgradable contract manually and with EVMPATCH.
3. patch an access control bug using EVMPATCH by writing a custom patch-template.

Task	Time (Minutes)			Confidence
	Median	Min	Max	Median (1-7)
Manual Integer Patches	47.50	35	78	6
Conversion	62.50	33	110	2.5
EVMPATCH Conversion	1.50	1	3	-
Patch Template	4.00	2	15	7

Conclusion


Vulnerability Detection
Automatic Analysis Tools
Vulnerability Disclosure
Attack Transactions
Developer
EVMPatch
Ethereum Blockchain
Vulnerability Report
Patch Tester
Transaction History
Vulnerable Contract Bytecode
Bytecode Rewriter
Patched Contract Bytecode
Patch Templates
Contract Deployment
Upgradable Contract