BLOSAFE
BLOCKCHAIN SCOPE

Wrapped Matic Token(MATIC) Smartcontract Security Audit Report

2022. 04

From SCOPE

https://blosafe.com

BLOSAFE
BLOCKCHAIN SCOPE

SCOPE

2022. 03

Document History

Date	Name	History
2022.03	Blosafe	Initial

1. Project outline

1.1. Purpose

The purpose of this inspection is to conduct a security audit on the [BnB] Smartcontract to discover potential hacking weaknesses, analyze the cause, and respond

1.2. Target

The subjects of this inspection are as follows.

No	Category	Addr	Memo
1	Smartcontract	0x7D1AfA7B718fb893dB30A3aBc0Cfc608AaCfeBB0	ETH Mainnet

1.3. Schedule

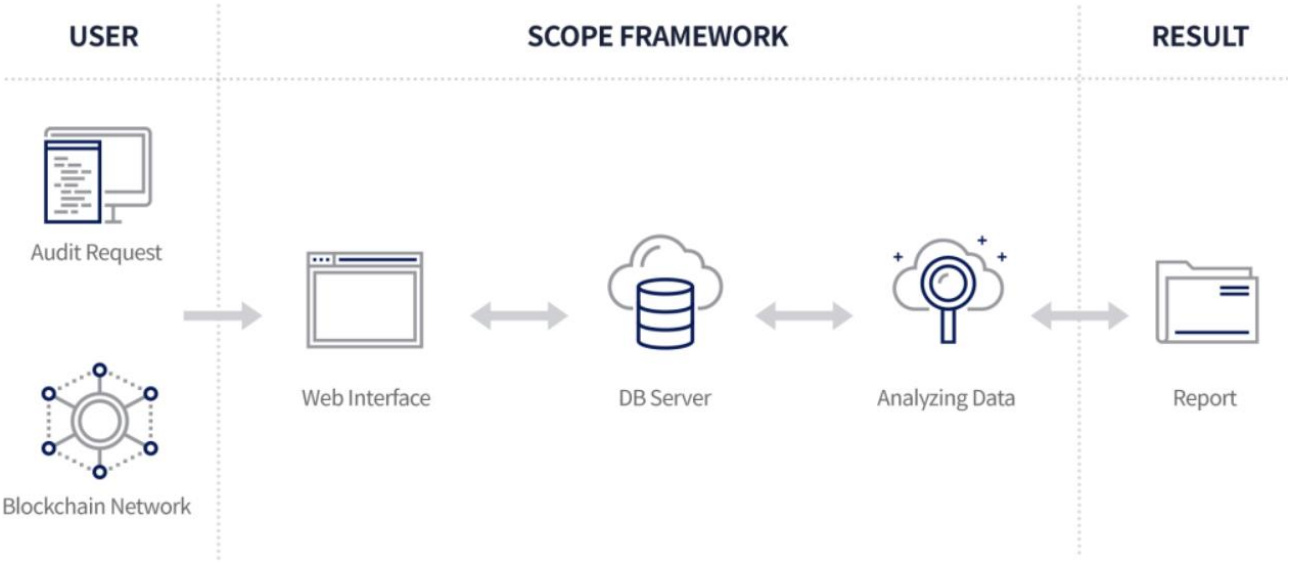
Work	Detail	Timeline	Memo
business consultation	Build Environment	1 day	
Audit	Smartcontract static auditing	2 days	
	Smartcontract Dynamic Auditing	3 days	
Report / review	Report	1 day	
	Review	1 day	

1.4. Environment

업무 구분	Name	Platform	Memo
Audit	Scope Audit	SaaS	

2. Process

2.1. Process Detail



2.2. Check List

No	Detector	What it Detects	Impact	external-function
1	`abiencoderv2-array`	Storage abiencoderv2 array	High	High
2	`array-by-reference`	Modifying storage array by value	High	High
3	`incorrect-shift`	The order of parameters in a shift instruction is incorrect.	High	High
4	`multiple-constructors`	Multiple constructor schemes	High	High
5	`name-reused`	Contract's name reused	High	High
6	`public-mappings-nested`	Public mappings with nested variables	High	High
7	`rtlo`	Right-To-Left-Override control character is used	High	High
8	`shadowing-state`	State variables shadowing	High	High
9	`suicidal`	Functions allowing anyone to destruct the contract	High	High

10	uninitialized-state	Uninitialized state variables	High	High
11	uninitialized-storage	Uninitialized storage variables	High	High
12	unprotected-upgrade	Unprotected upgradeable contract	High	High
13	arbitrary-send	Functions that send Ether to arbitrary destinations	High	Medium
14	controlled-array-length	Tainted array length assignment	High	Medium
15	controlled-delegatecall	Controlled delegatecall destination	High	Medium
16	delegatecall-loop	Payable functions using delegatecall inside a loop	High	Medium
17	msg-value-loop	msg.value inside a loop	High	Medium
18	reentrancy-eth	Reentrancy vulnerabilities (theft of ethers)	High	Medium
19	storage-array	Signed storage integer array compiler bug	High	Medium
20	unchecked-transfer	Unchecked tokens transfer	High	Medium
21	weak-prng	Weak PRNG	High	Medium
22	enum-conversion	Detect dangerous enum conversion	Medium	High
23	erc20-interface	Incorrect ERC20 interfaces	Medium	High
24	erc721-interface	Incorrect ERC721 interfaces	Medium	High
25	incorrect-equality	Dangerous strict equalities	Medium	High
26	locked-ether	Contracts that lock ether	Medium	High
27	mapping-deletion	Deletion on mapping containing a structure	Medium	High
28	shadowing-abstract	State variables shadowing from abstract contracts	Medium	High
29	tautology	Tautology or contradiction	Medium	High
30	write-after-write	Unused write	Medium	High
31	boolean-cst	Misuse of Boolean constant	Medium	Medium

SCOPE

32	constant-function-asm	Constant functions using assembly code	Medium	Medium
33	constant-function-state	Constant functions changing the state	Medium	Medium
34	divide-before-multiply	Imprecise arithmetic operations order	Medium	Medium
35	reentrancy-no-eth	Reentrancy vulnerabilities (no theft of ethers)	Medium	Medium
36	reused-constructor	Reused base constructor	Medium	Medium
37	tx-origin	Dangerous usage of tx.origin	Medium	Medium
38	unchecked-lowlevel	Unchecked low-level calls	Medium	Medium
39	unchecked-send	Unchecked send	Medium	Medium
40	uninitialized-local	Uninitialized local variables	Medium	Medium
41	unused-return	Unused return values	Medium	Medium
42	incorrect-modifier	Modifiers that can return the default value	Low	High
43	shadowing-builtin	Built-in symbol shadowing	Low	High
44	shadowing-local	Local variables shadowing	Low	High
45	uninitialized-fptr-cst	Uninitialized function pointer calls in constructors	Low	High
46	variable-scope	Local variables used prior their declaration	Low	High
47	void-cst	Constructor called not implemented	Low	High
48	calls-loop	Multiple calls in a loop	Low	Medium
49	events-access	Missing Events Access Control	Low	Medium
50	events-maths	Missing Events Arithmetic	Low	Medium
51	incorrect-unary	Dangerous unary expressions	Low	Medium
52	missing-zero-check	Missing Zero Address Validation	Low	Medium
53	reentrancy-benign	Benign reentrancy vulnerabilities	Low	Medium

SCOPE

54	reentrancy-events	Reentrancy vulnerabilities leading to out-of-order Events	Low	Medium
55	timestamp	Dangerous usage of block.timestamp	Low	Medium
56	assembly	Assembly usage	Informational	High
57	assert-state-change	Assert state change	Informational	High
58	boolean-equal	Comparison to boolean constant	Informational	High
59	deprecated-standards	Deprecated Solidity Standards	Informational	High
60	erc20-indexed	Un-indexed ERC20 event parameters	Informational	High
61	function-init-state	Function initializing state variables	Informational	High
62	low-level-calls	Low level calls	Informational	High
63	missing-inheritance	Missing inheritance	Informational	High
64	naming-convention	Conformity to Solidity naming conventions	Informational	High
65	pragma	If different pragma directives are used	Informational	High
66	redundant-statements	Redundant statements	Informational	High
67	solc-version	Incorrect Solidity version	Informational	High
68	unimplemented-functions	Unimplemented functions	Informational	High
69	unused-state	Unused state variables	Informational	High
70	costly-loop	Costly operations in a loop	Informational	Medium
71	dead-code	Functions that are not used	Informational	Medium
72	reentrancy-unlimited-gas	Reentrancy vulnerabilities through send and transfer	Informational	Medium
73	similar-names	Variable names are too similar	Informational	Medium
74	too-many-digits	Conformance to numeric notation best practices	Informational	Medium
75	constable-states	State variables that could be declared constant	Optimization	High

SCOPE

76	external-function	Public function that could be declared external	Optimization	High

SCOPE

Summary of results

2.3. Result

[Passed]

[MATIC] As a result of the Smartcontract security audit, a total of 13 vulnerabilities were found, among which 0 vulnerabilities of 'high', 0 of 'medium' vulnerabilities, 7 of 'low' vulnerabilities, and 6 of 'information' ratings were found.

3. Detailed results

3.1. Smartcontract

```
/**
 *Submitted for verification at Etherscan.io on 2019-04-20
*/

pragma solidity 0.5.2;

// File: openzeppelin-solidity/contracts/token/ERC20/IERC20.sol

/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
interface IERC20 {
    function transfer(address to, uint256 value) external returns (bool);

    function approve(address spender, uint256 value) external returns (bool);

    function transferFrom(address from, address to, uint256 value) external returns (bool);

    function totalSupply() external view returns (uint256);

    function balanceOf(address who) external view returns (uint256);
```

```
    function allowance(address owner, address spender) external view returns (uint256);

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Approval(address indexed owner, address indexed spender, uint256 value);
}

// File: openzeppelin-solidity/contracts/math/SafeMath.sol

/**
 * @title SafeMath
 * @dev Unsigned math operations with safety checks that revert on error
 */
library SafeMath {
    /**
    * @dev Multiplies two unsigned integers, reverts on overflow.
    */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }
```

```
        uint256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
    * @dev Integer division of two unsigned integers truncating the quotient, reverts on division by zero.
    */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }

    /**
    * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend is greater than minuend).
    */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a);
```

```
        uint256 c = a - b;

        return c;
    }

    /**
    * @dev Adds two unsigned integers, reverts on overflow.
    */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a);

        return c;
    }

    /**
    * @dev Divides two unsigned integers and returns the remainder (unsigned integer modulo),
    * reverts when dividing by zero.
    */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}
```


SCOPE

```
// File: openzeppelin-solidity/contracts/token/ERC20/ERC20.sol


/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
 * Originally based on code by FirstBlood:
 * https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.sol
 *
 * This implementation emits additional Approval events, allowing applications to reconstruct the allowance status for
 * all accounts just by listening to said events. Note that this isn't required by the specification, and other
 * compliant implementations may not do it.
 */
contract ERC20 is IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowed;

    uint256 private _totalSupply;

    /**
```


SCOPE

```
* @dev Total number of tokens in existence
*/
function totalSupply() public view returns (uint256) {
    return _totalSupply;
}

/**
* @dev Gets the balance of the specified address.
* @param owner The address to query the balance of.
* @return An uint256 representing the amount owned by the passed address.
*/
function balanceOf(address owner) public view returns (uint256) {
    return _balances[owner];
}

/**
 * @dev Function to check the amount of tokens that an owner allowed to a spender.
 * @param owner address The address which owns the funds.
 * @param spender address The address which will spend the funds.
 * @return A uint256 specifying the amount of tokens still available for the spender.
 */
function allowance(address owner, address spender) public view returns (uint256) {
    return _allowed[owner][spender];
}
```

```
/**
* @dev Transfer token for a specified address
* @param to The address to transfer to.
* @param value The amount to be transferred.
*/
function transfer(address to, uint256 value) public returns (bool) {
    _transfer(msg.sender, to, value);
    return true;
}

/**
 * @dev Approve the passed address to spend the specified amount of tokens on behalf of msg.sender.
 * Beware that changing an allowance with this method brings the risk that someone may use both the old
 * and the new allowance by unfortunate transaction ordering. One possible solution to mitigate this
 * race condition is to first reduce the spender's allowance to 0 and set the desired value afterwards:
 * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
 * @param spender The address which will spend the funds.
 * @param value The amount of tokens to be spent.
 */
function approve(address spender, uint256 value) public returns (bool) {
    require(spender != address(0));

    _allowed[msg.sender][spender] = value;
    emit Approval(msg.sender, spender, value);
```


SCOPE

```
        return true;
    }

    /**
     * @dev Transfer tokens from one address to another.
     * Note that while this function emits an Approval event, this is not required as per the specification,
     * and other compliant implementations may not emit the event.
     * @param from address The address which you want to send tokens from
     * @param to address The address which you want to transfer to
     * @param value uint256 the amount of tokens to be transferred
     */
    function transferFrom(address from, address to, uint256 value) public returns (bool) {
        _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
        _transfer(from, to, value);
        emit Approval(from, msg.sender, _allowed[from][msg.sender]);
        return true;
    }

    /**
     * @dev Increase the amount of tokens that an owner allowed to a spender.
     * approve should be called when allowed_[_spender] == 0. To increment
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * Emits an Approval event.
```


SCOPE

```
 * @param spender The address which will spend the funds.
 * @param addedValue The amount of tokens to increase the allowance by.
 */
function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
    require(spender != address(0));

    _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
    emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
    return true;
}

/**
 * @dev Decrease the amount of tokens that an owner allowed to a spender.
 * approve should be called when allowed_[_spender] == 0. To decrement
 * allowed value is better to use this function to avoid 2 calls (and wait until
 * the first transaction is mined)
 * From MonolithDAO Token.sol
 * Emits an Approval event.
 * @param spender The address which will spend the funds.
 * @param subtractedValue The amount of tokens to decrease the allowance by.
 */
function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
    require(spender != address(0));

    _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(subtractedValue);
```


SCOPE

```
        emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
        return true;
    }

    /**
    * @dev Transfer token for a specified addresses
    * @param from The address to transfer from.
    * @param to The address to transfer to.
    * @param value The amount to be transferred.
    */
    function _transfer(address from, address to, uint256 value) internal {
        require(to != address(0));

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
    }

    /**
     * @dev Internal function that mints an amount of the token and assigns it to
     * an account. This encapsulates the modification of balances such that the
     * proper events are emitted.
     * @param account The account that will receive the created tokens.
     * @param value The amount that will be created.
     */
```

```
function _mint(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
}

/**
 * @dev Internal function that burns an amount of the token of a given
 * account.
 * @param account The account whose tokens will be burnt.
 * @param value The amount that will be burnt.
 */
function _burn(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
}

/**
 * @dev Internal function that burns an amount of the token of a given
 * account, deducting from the sender's allowance for said account. Uses the
```

```
     * internal burn function.
     * Emits an Approval event (reflecting the reduced allowance).
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
    function _burnFrom(address account, uint256 value) internal {
        _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(value);
        _burn(account, value);
        emit Approval(account, msg.sender, _allowed[account][msg.sender]);
    }
}

// File: openzeppelin-solidity/contracts/access/Roles.sol

/**
 * @title Roles
 * @dev Library for managing addresses assigned to a Role.
 */
library Roles {
    struct Role {
        mapping (address => bool) bearer;
    }

    /**
     * @dev give an account access to this role
```

```
 */
function add(Role storage role, address account) internal {
    require(account != address(0));
    require(!has(role, account));

    role.bearer[account] = true;
}

/**
 * @dev remove an account's access to this role
 */
function remove(Role storage role, address account) internal {
    require(account != address(0));
    require(has(role, account));

    role.bearer[account] = false;
}

/**
 * @dev check if an account has this role
 * @return bool
 */
function has(Role storage role, address account) internal view returns (bool) {
    require(account != address(0));
    return role.bearer[account];
```

```
    }
}

// File: openzeppelin-solidity/contracts/access/roles/PauserRole.sol

contract PauserRole {
    using Roles for Roles.Role;

    event PauserAdded(address indexed account);
    event PauserRemoved(address indexed account);

    Roles.Role private _pausers;

    constructor () internal {
        _addPauser(msg.sender);
    }

    modifier onlyPauser() {
        require(isPauser(msg.sender));
        _;
    }

    function isPauser(address account) public view returns (bool) {
        return _pausers.has(account);
    }
```

```
    function addPauser(address account) public onlyPauser {
        _addPauser(account);
    }

    function renouncePauser() public {
        _removePauser(msg.sender);
    }

    function _addPauser(address account) internal {
        _pausers.add(account);
        emit PauserAdded(account);
    }

    function _removePauser(address account) internal {
        _pausers.remove(account);
        emit PauserRemoved(account);
    }
}

// File: openzeppelin-solidity/contracts/lifecycle/Pausable.sol

/**
 * @title Pausable
 * @dev Base contract which allows children to implement an emergency stop mechanism.
```

```
 */
contract Pausable is PauserRole {
    event Paused(address account);
    event Unpaused(address account);

    bool private _paused;

    constructor () internal {
        _paused = false;
    }

    /**
     * @return true if the contract is paused, false otherwise.
     */
    function paused() public view returns (bool) {
        return _paused;
    }

    /**
     * @dev Modifier to make a function callable only when the contract is not paused.
     */
    modifier whenNotPaused() {
        require(!_paused);
        _;
    }
```

```
    /**
     * @dev Modifier to make a function callable only when the contract is paused.
     */
    modifier whenPaused() {
        require(_paused);
        _;
    }

    /**
     * @dev called by the owner to pause, triggers stopped state
     */
    function pause() public onlyPauser whenNotPaused {
        _paused = true;
        emit Paused(msg.sender);
    }

    /**
     * @dev called by the owner to unpause, returns to normal state
     */
    function unpause() public onlyPauser whenPaused {
        _paused = false;
        emit Unpaused(msg.sender);
    }
}
```

```
// File: openzeppelin-solidity/contracts/token/ERC20/ERC20Pausable.sol


/**
 * @title Pausable token
 * @dev ERC20 modified with pausable transfers.
 **/
contract ERC20Pausable is ERC20, Pausable {
    function transfer(address to, uint256 value) public whenNotPaused returns (bool) {
        return super.transfer(to, value);
    }

    function transferFrom(address from, address to, uint256 value) public whenNotPaused returns (bool) {
        return super.transferFrom(from, to, value);
    }

    function approve(address spender, uint256 value) public whenNotPaused returns (bool) {
        return super.approve(spender, value);
    }

    function increaseAllowance(address spender, uint addedValue) public whenNotPaused returns (bool success) {
        return super.increaseAllowance(spender, addedValue);
    }
```

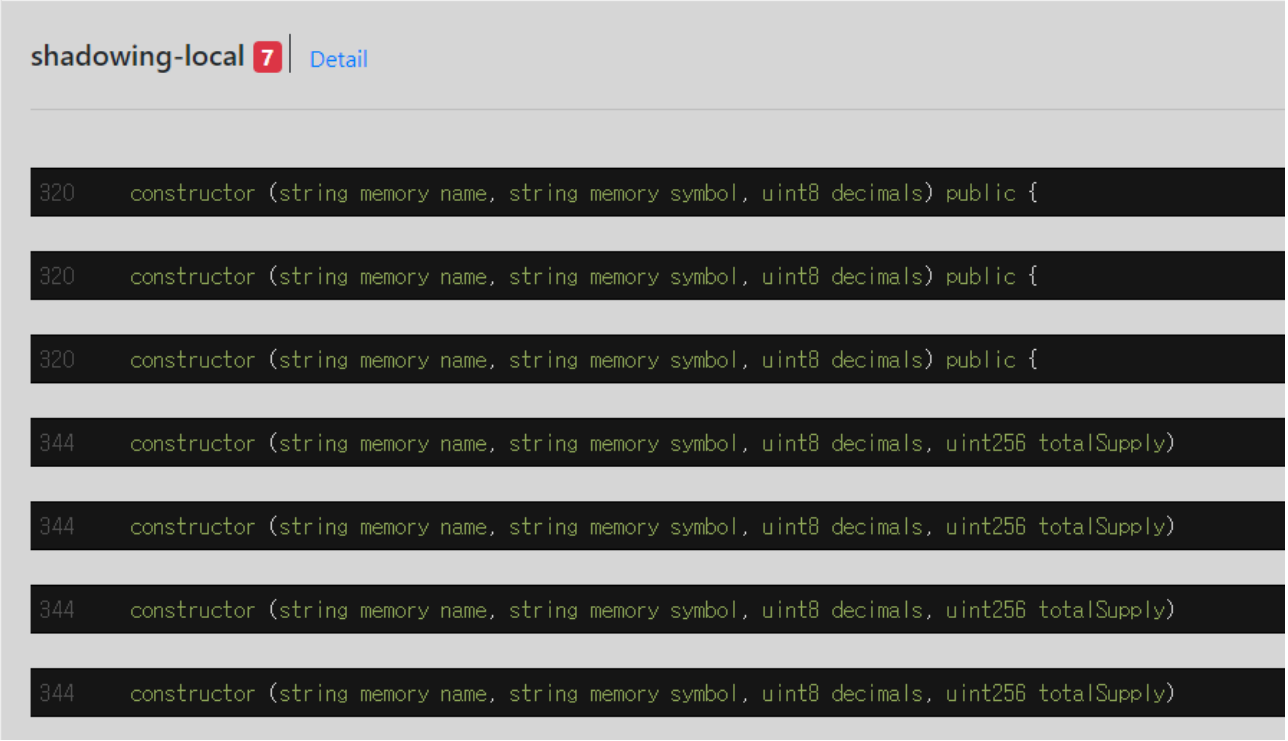
```
    function decreaseAllowance(address spender, uint subtractedValue) public whenNotPaused returns (bool success) {
        return super.decreaseAllowance(spender, subtractedValue);
    }
}

// File: openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol

/**
 * @title ERC20Detailed token
 * @dev The decimals are only for visualization purposes.
 * All the operations are done using the smallest and indivisible token unit,
 * just as on Ethereum all the operations are done in wei.
 */
contract ERC20Detailed is IERC20 {
    string private _name;
    string private _symbol;
    uint8 private _decimals;

    constructor (string memory name, string memory symbol, uint8 decimals) public {
        _name = name;
        _symbol = symbol;
        _decimals = decimals;
    }
```

```
    /**
     * @return the name of the token.
     */
    function name() public view returns (string memory) {
        return _name;
    }

    /**
     * @return the symbol of the token.
     */
    function symbol() public view returns (string memory) {
        return _symbol;
    }

    /**
     * @return the number of decimals of the token.
     */
    function decimals() public view returns (uint8) {
        return _decimals;
    }
}

// File: contracts/MaticToken.sol

contract MaticToken is ERC20Pausable, ERC20Detailed {
```

```
    constructor (string memory name, string memory symbol, uint8 decimals, uint256 totalSupply)
    public
    ERC20Detailed (name, symbol, decimals) {
        _mint(msg.sender, totalSupply);
    }
}}
```

3.2. Vulnerability

shadowing-local



onfiguration

- Check: `shadowing-local`
- Severity: `Low`
- Confidence: `High`

Description

Detection of shadowing using local variables.

Exploit Scenario:

```
pragma solidity ^0.4.24;

contract Bug {
    uint owner;

    function sensitive_function(address owner) public {
        // ...
        require(owner == msg.sender);
    }

    function alternate_sensitive_function() public {
        address owner = msg.sender;
        // ...
        require(owner == msg.sender);
    }
}
```

`sensitive_function.owner` shadows `Bug.owner`. As a result, the use of `owner` in `sensitive_function` might be incorrect.

Recommendation

Rename the local variables that shadow another component.

external-function

external-function 11 | Detail

```
86    function totalSupply() public view returns (uint256) {
87        return _totalSupply;
88    }
```

```
91    function balanceOf(address owner) public view returns (uint256) {
92        return _balances[owner];
93    }
```

```
96    function allowance(address owner, address spender) public view returns (uint256) {
97        return _allowed[owner][spender];
98    }
```

```
228   function addPauser(address account) public onlyPauser {
229       _addPauser(account);
230   }
```

```
232   function renouncePauser() public {
233       _removePauser(msg.sender);
234   }
```

```
260   function paused() public view returns (bool) {
261       return _paused;
262   }
```

```
277   function pause() public onlyPauser whenNotPaused {
278       _paused = true;
279       emit Paused(msg.sender);
280   }
```

```
283   function unpause() public onlyPauser whenPaused {
284       _paused = false;
285       emit Unpaused(msg.sender);
286   }
```

```
327   function name() public view returns (string memory) {
328       return _name;
329   }
```

```
332   function symbol() public view returns (string memory) {
333       return _symbol;
334   }
```

```
337   function decimals() public view returns (uint8) {
338       return _decimals;
339   }
```

Configuration

- Check: `external-function`
- Severity: `Optimization`
- Confidence: `High`

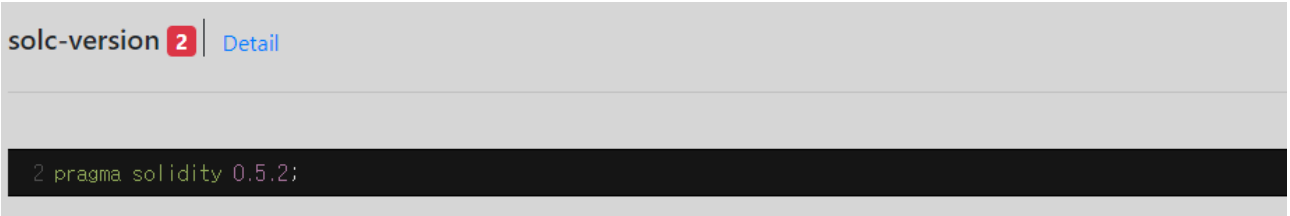
Description

`public` functions that are never called by the contract should be declared `external` to save gas.

Recommendation

Use the `external` attribute for functions never called from the contract.

solc-version



Configuration

- Check: `solc-version`
- Severity: `Informational`
- Confidence: `High`

Description

`solc` frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks. We also recommend avoiding complex `pragma` statement.

Recommendation

Deploy with any of the following Solidity versions:

- 0.5.16 - 0.5.17
- 0.6.11 - 0.6.12
- 0.7.5 - 0.7.6 Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

dead-code

dead-code 5 Detail

```
160    function _burn(address account, uint256 value) internal {
161        require(account != address(0));
162
163        _totalSupply = _totalSupply.sub(value);
164        _balances[account] = _balances[account].sub(value);
165        emit Transfer(account, address(0), value);
166    }
```

```
169    function _burnFrom(address account, uint256 value) internal {
170        _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(value);
171        _burn(account, value);
172        emit Approval(account, msg.sender, _allowed[account][msg.sender]);
173    }
```

```
44     function div(uint256 a, uint256 b) internal pure returns (uint256) {
45                 require(b > 0);
46         uint256 c = a / b;
47
48         return c;
49     }
```

```
68     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
69         require(b != 0);
70         return a % b;
71     }
```

```
32     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
33                             if (a == 0) {
34             return 0;
35         }
36
37         uint256 c = a * b;
38         require(c / a == b);
39
40         return c;
41     }
```

Configuration

- Check: `dead-code`
- Severity: `Informational`
- Confidence: `Medium`

Description

Functions that are not sued.

Exploit Scenario:

```
contract Contract{
    function dead_code() internal() {}
}
```

`dead_code` is not used in the contract, and make the code's review more difficult.

Recommendation

Remove unused functions.