MEDIUM

SWC-000

## Incorrect ERC20 implementation

Contract "BasicToken" looks like its trying to implement the ERC20 standard, but its missing a required event with signature "event Approval(address indexed, address indexed, uint256)"

Source file

/contracts/enatoken.sol

Locations

```
* @dev Basic version of StandardToken, with no allowances.
*/
contract BasicToken is ERC20Basic {
using SafeMath for uint256;

mapping(address => uint256) internal balances;

/**
* @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) {
require(_to != address(0) && _to != address(this));

// SafeMath.sub will throw if there is not enough balance.
balances[msg.sender] = balances[msg.sender].sub(_value);
balances[_to] = balances[_to].add(_value);
emit Transfer(msg.sender, _to, _value);
return true;
}

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

/**
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "transferOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
* @param newOwner The address to transfer ownership to.
*/
function transferOwnership(address newOwner) public onlyOwner {
require(newOwner != address(0));
emit OwnershipTransferred(owner, newOwner);
owner = newOwner;
}
}
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "balanceOf" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`/contracts/enatoken.sol`

Locations

```
uint256 public totalSupply;

function balanceOf(address who) public view returns (uint256);

function transfer(address to, uint256 value) public returns (bool);
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`/contracts/enatoken.sol`

Locations

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

function transfer(address to, uint256 value) public returns (bool);

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

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`/contracts/enatoken.sol`

Locations

```
* @param _value The amount to be transferred.
*/
function transfer(address _to, uint256 _value) public returns (bool) {
require(_to != address(0) && _to != address(this));

// SafeMath.sub will throw if there is not enough balance.
balances[msg.sender] = balances[msg.sender].sub(_value);
balances[_to] = balances[_to].add(_value);
emit Transfer(msg.sender, _to, _value);
return true;
}

/**
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "balanceOf" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
* @return An uint256 representing the amount owned by the passed address.
*/
function balanceOf(address _owner) public view returns (uint256 balance) {
return balances[_owner];
}
}
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "allowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
*/
contract ERC20 is ERC20Basic {
function allowance(address owner, address spender)
public
view
returns (uint256);

function transferFrom(
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
returns (uint256);

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

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

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "approve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
) public returns (bool);

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

event Approval(
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
* @param _value uint256 the amount of tokens to be transferred
*/
function transferFrom(
address _from,
address _to,
uint256 _value
) public returns (bool) {
require(_to != address(0) && _to != address(this));

uint256 _allowance = allowed[_from][msg.sender];

// Check is not needed because sub(_allowance, _value) will already throw if this condition is not met
// require (_value <= _allowance);

balances[_from] = balances[_from].sub(_value);
balances[_to] = balances[_to].add(_value);
allowed[_from][msg.sender] = _allowance.sub(_value);
emit Transfer(_from, _to, _value);
return true;
}

/**
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "allowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
* @return A uint256 specifying the amount of tokens still available for the spender.
*/
function allowance(address _owner, address _spender)
public
view
returns (uint256 remaining)
{
return allowed[_owner][_spender];
}

/**
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "increaseApproval" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
* From ENAToken
*/
function increaseApproval(address _spender, uint256 _addedValue)
public
returns (bool success)
{
allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(
_addedValue
);
emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
return true;
}

function decreaseApproval(address _spender, uint256 _subtractedValue)
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "decreaseApproval" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
}

function decreaseApproval(address _spender, uint256 _subtractedValue)
public
returns (bool success)
{
uint256 oldValue = allowed[msg.sender][_spender];
if (_subtractedValue > oldValue) {
allowed[msg.sender][_spender] = 0;
} else {
allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
}
emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
return true;
}
}
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "burn" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
* @param _value The amount of token to be burned.
*/
function burn(uint256 _value) public {
require(_value > 0);
require(_value <= balances[msg.sender]);
// no need to require value <= totalSupply, since that would imply the
// sender's balance is greater than the totalSupply, which *should* be an assertion failure

address burner = msg.sender;
balances[burner] = balances[burner].sub(_value);
totalSupply = totalSupply.sub(_value);
emit Burn(burner, _value);
emit Transfer(burner, address(0), _value);
}
}
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "transferAnyERC20Token" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/enatoken.sol

Locations

```
}

function transferAnyERC20Token(
address _tokenAddress,
address _to,
uint256 _amount
) public onlyOwner {
ERC20(_tokenAddress).transfer(_to, _amount);
}
}
```

LOW

SWC-107

## A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Source file

/contracts/enatoken.sol

Locations

```
uint256 _amount
) public onlyOwner {
ERC20(_tokenAddress).transfer(_to, _amount);
}
}
```

LOW

SWC-107

## A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Source file

/contracts/enatoken.sol

Locations

```
TokenRecipient spender = TokenRecipient(_spender);
if (approve(_spender, _value)) {
spender.receiveApproval(msg.sender, _value, _extraData);
return true;
}
```

LOW

SWC-110

## An assertion violation was triggered.

It is possible to cause an assertion violation. Note that Solidity assert() statements should only be used to check invariants. Review the transaction trace generated for this issue and either make sure your program logic is correct, or use require() instead of assert() if your goal is to constrain user inputs or enforce preconditions. Remember to validate inputs from both callers (for instance, via passed arguments) and callees (for instance, via return values).

Source file

`/contracts/enatoken.sol`

Locations

```

function sub(uint256 a, uint256 b) internal pure returns (uint256) {
assert(b <= a);
return a - b;
}
```