September 1, 2025

Express Audit Report for

Smart_DeFi_NetWork

De.Fi

De.Fi

Project Summary

Project Name	Smart_DeFi_NetWork
Address	**0xd341197ee1171d30c0b1685b521c140a6299c200**
Network	56

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	**function BeCome_Owner**(**address** Up) **external** {DC(Up);} **function DC**(**address** Up) **private** { require(I_C(_msgSender()) == false, "Just Wallet"); require(Up != **address**(0), "Dont Enter address 0"); require(KW[Up].XI != 2, " Upline Has 2 Directs "); require(_msgSender() != Up, " Dont Enter Your Address "); require(!DX(_msgSender()), " You Are An Owner "); require(DX(Up), " Upline Not Exist "); require(Agreement_[_msgSender()] == true, " Write Agreement"); require(Owner_Is_My_Line(Founder_Wallet , Up) == true, "UpLines Are Not Imported"); require(Waiting == false, " Processing "); Waiting = true; stableCoin.safeTransferFrom(_msgSender(), **address**(this), 100 * 10**18); IERC20(stableCoin).transfer(smart_Gift, 5 * 10**18); VV[JK] = _msgSender(); JK++; Node **memory** owner = Node({ id: JK, AL: 0, AR: 0, LT: 0, RT: 0, XI: 0 , YY: KW[Up].XI == 0 ? false : true, UP: Up, PO: **address**(0), QO: **address**(0)}); KW[_msgSender()] = owner; JJ[newMember] = _msgSender(); DZ++;newMember++; **if** (KW[Up].XI == 0) {KW[Up].LT++; KW[Up].AL++; KW[Up].PO = _msgSender();} **else** {KW[Up].RT++; KW[Up].AR++; KW[Up].QO = _msgSender();} KW[Up].XI++; Waiting = false;}
Location	Smart_DeFi_NetWork.DC(address) (Smart_DeFi_NetWork.sol#33) compares to a boolean constant: -require(bool,string)(I_C(_msgSender()) == false,Just Wallet) (Smart_DeFi_NetWork.sol#33)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	function BeCome_Owner(address Up) external {DC(Up);} function DC(address Up) private { require(I_C(_msgSender()) == false, "Just Wallet"); require(Up != address(0), "Dont Enter address 0"); require(KW[Up].XI != 2, " Upline Has 2 Directs "); require(_msgSender() != Up, " Dont Enter Your Address "); require(!DX(_msgSender()), " You Are An Owner "); require(DX(Up), " Upline Not Exist "); require(Agreement_[_msgSender()] == true, " Write Agreement"); require(Owner_Is_My_Line(Founder_Wallet , Up) == true, "UpLines Are Not Imported"); require(Waiting == false, " Processing "); Waiting = true; stableCoin.safeTransferFrom(_msgSender(), address(this), 100 * 10**18); IERC20(stableCoin).transfer(smart_Gift, 5 * 10**18); VV[JK] = _msgSender(); JK++; Node memory owner = Node({ id: JK, AL: 0, AR: 0, LT: 0, RT: 0, XI: 0 , YY: KW[Up].XI == 0 ? false : true, UP: Up, PO: address(0), QO: address(0)}); KW[_msgSender()] = owner; JJ[newMember] = _msgSender(); DZ++;newMember++; if (KW[Up].XI == 0) {KW[Up].LT++; KW[Up].AL++; KW[Up].PO = _msgSender();} else {KW[Up].RT++; KW[Up].AR++; KW[Up].QO = _msgSender();} KW[Up].XI++; Waiting = false;}
Location	Smart_DeFi_NetWork.DC(address) (Smart_DeFi_NetWork.sol#33) compares to a boolean constant: -require(bool,string)(Waiting == false, Processing) (Smart_DeFi_NetWork.sol#33)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	**function BeCome_Owner**(**address** Up) **external** {DC(Up);} **function DC**(**address** Up) **private** { require(I_C(_msgSender()) == false, "Just Wallet"); require(Up != **address**(0), "Dont Enter address 0"); require(KW[Up].XI != 2, " Upline Has 2 Directs "); require(_msgSender() != Up, " Dont Enter Your Address "); require(!DX(_msgSender()), " You Are An Owner "); require(DX(Up), " Upline Not Exist "); require(Agreement_[_msgSender()] == true, " Write Agreement"); require(Owner_Is_My_Line(Founder_Wallet , Up) == true, "UpLines Are Not Imported"); require(Waiting == false, " Processing "); Waiting = true; stableCoin.safeTransferFrom(_msgSender(), **address**(this), 100 * 10**18); IERC20(stableCoin).transfer(smart_Gift, 5 * 10**18); VV[JK] = _msgSender(); JK++; Node **memory** owner = Node({ id: JK, AL: 0, AR: 0, LT: 0, RT: 0, XI: 0 , YY: KW[Up].XI == 0 ? false : true, UP: Up, PO: **address**(0), QO: **address**(0)}); KW[_msgSender()] = owner; JJ[newMember] = _msgSender(); DZ++;newMember++; **if** (KW[Up].XI == 0) {KW[Up].LT++; KW[Up].AL++; KW[Up].PO = _msgSender();} **else** {KW[Up].RT++; KW[Up].AR++; KW[Up].QO = _msgSender();} KW[Up].XI++; Waiting = false;}
Location	Smart_DeFi_NetWork.DC(address) (Smart_DeFi_NetWork.sol#33) compares to a boolean constant: -require(bool,string)(Agreement_[_msgSender()] == true, Write Agreement) (Smart_DeFi_NetWork.sol#33)

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function BeCome_Owner**(**address** Up) **external** {DC(Up);} **function DC**(**address** Up) **private** { require(I_C(_msgSender()) == false, "Just Wallet"); require(Up != **address**(0), "Dont Enter address 0"); require(KW[Up].XI != 2, " Upline Has 2 Directs "); require(_msgSender() != Up, " Dont Enter Your Address "); require(!DX(_msgSender()), " You Are An Owner "); require(DX(Up), " Upline Not Exist "); require(Agreement_[_msgSender()] == true, " Write Agreement"); require(Owner_Is_My_Line(Founder_Wallet , Up) == true, "UpLines Are Not Imported"); require(Waiting == false, " Processing "); Waiting = true; stableCoin.safeTransferFrom(_msgSender(), **address**(this), 100 * 10**18); IERC20(stableCoin).transfer(smart_Gift, 5 * 10**18); VV[JK] = _msgSender(); JK++; Node **memory** owner = Node({ id: JK, AL: 0, AR: 0, LT: 0, RT: 0, XI: 0 , YY: KW[Up].XI == 0 ? false : true, UP: Up, PO: **address**(0), QO: **address**(0)}); KW[_msgSender()] = owner; JJ[newMember] = _msgSender(); DZ++;newMember++; **if** (KW[Up].XI == 0) {KW[Up].LT++; KW[Up].AL++; KW[Up].PO = _msgSender();} **else** {KW[Up].RT++; KW[Up].AR++; KW[Up].QO = _msgSender();} KW[Up].XI++; Waiting = false;}
Location	Smart_DeFi_NetWork.DC(address) (Smart_DeFi_NetWork.sol#33) compares to a boolean constant: -require(bool,string) (Owner_Is_My_Line(Founder_Wallet,Up) == true,UpLines Are Not Imported) (Smart_DeFi_NetWork.sol#33)

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function Reward() external** {DH();} **function DH() private** { require(I_C(_msgSender()) == false, "Just Wallet"); require(Owner_All_Point(_msgSender()) > 0, "Just NetWorker"); require(block.timestamp > time + 1 hours, " Reward Time Has Not Come "); ZB(); require(ZI() > 0, " Total Point Is 0 "); require(Waiting == false, " Processing "); Waiting = true ; ZL = ZI(); JY = _msgSender(); **uint256** ZO = ZK(); ZM = ZO; RCr++; VPL[RCr] = ZO; **uint256** D_T = ((DZ * rewardFee * 10**18) / 2); **for** (**uint24** i = 0; i < DJ; i++) { Node **memory** ZN = KW[JL[i]]; **uint32** UT = ZH(JL[i]); **if** (ZN.LT == UT) {ZN.LT = 0; ZN.RT -= UT;} **else if** (ZN.RT == UT) {ZN.LT -= UT; ZN.RT = 0;} **else** { **if** (ZN.LT < ZN.RT) {ZN.RT -= ZN.LT; ZN.LT = 0;} **else** {ZN.LT -= ZN.RT; ZN.RT = 0;}} KW[JL[i]] = ZN; **if** (Owner_All_Point(JL[i]) < 100) { **if** (UT * ZO > stableCoin.balanceOf(**address**(this))) { stableCoin.safeTransfer(JL[i], stableCoin.balanceOf(**address**(this))); } **else** { stableCoin.safeTransfer(JL[i], UT * ZO); }} **else** { **if** (((UT * ZO * 9) / 10) > stableCoin.balanceOf(**address**(this))) {stableCoin.safeTransfer(JL[i], stableCoin.balanceOf(**address**(this))); } **else** { stableCoin.safeTransfer(JL[i], ((UT * ZO * 9) / 10)); } } } **if** (D_T <= stableCoin.balanceOf(**address**(this))) {stableCoin.safeTransfer(_msgSender(), D_T);} stableCoin.safeTransfer(smart_Bank, stableCoin.balanceOf(**address**(this))); time = block.timestamp; DZ = 0; newMember = 0; LZ = 0; DJ = 0; Waiting = false ;}
Location	Smart_DeFi_NetWork.DH() (Smart_DeFi_NetWork.sol#34) compares to a boolean constant: -require(bool,string)(Waiting == false, Processing) (Smart_DeFi_NetWork.sol#34)

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function Reward() external** {DH();} **function DH() private** { require(I_C(_msgSender()) == false, "Just Wallet"); require(Owner_All_Point(_msgSender()) > 0, "Just NetWorker"); require(block.timestamp > time + 1 hours, " Reward Time Has Not Come "); ZB(); require(ZI() > 0, " Total Point Is 0 "); require(Waiting == false, " Processing "); Waiting = true ; ZL = ZI(); JY = _msgSender(); **uint256** ZO = ZK(); ZM = ZO; RCr++; VPL[RCr] = ZO; **uint256** D_T = ((DZ * rewardFee * 10**18) / 2); **for** (**uint24** i = 0; i < DJ; i++) { Node **memory** ZN = KW[JL[i]]; **uint32** UT = ZH(JL[i]); **if** (ZN.LT == UT) {ZN.LT = 0; ZN.RT -= UT;} **else if** (ZN.RT == UT) {ZN.LT -= UT; ZN.RT = 0;} **else** { **if** (ZN.LT < ZN.RT) {ZN.RT -= ZN.LT; ZN.LT = 0;} **else** {ZN.LT -= ZN.RT; ZN.RT = 0;}} KW[JL[i]] = ZN; **if** (Owner_All_Point(JL[i]) < 100) { **if** (UT * ZO > stableCoin.balanceOf(**address**(this))) { stableCoin.safeTransfer(JL[i], stableCoin.balanceOf(**address**(this))); } **else** { stableCoin.safeTransfer(JL[i], UT * ZO); }} **else** { **if** (((UT * ZO * 9) / 10) > stableCoin.balanceOf(**address**(this))) {stableCoin.safeTransfer(JL[i], stableCoin.balanceOf(**address**(this))); } **else** { stableCoin.safeTransfer(JL[i], ((UT * ZO * 9) / 10)); } } } **if** (D_T <= stableCoin.balanceOf(**address**(this))) {stableCoin.safeTransfer(_msgSender(), D_T);} stableCoin.safeTransfer(smart_Bank, stableCoin.balanceOf(**address**(this))); time = block.timestamp; DZ = 0; newMember = 0; LZ = 0; DJ = 0; Waiting = false ;}
Location	Smart_DeFi_NetWork.DH() (Smart_DeFi_NetWork.sol#34) compares to a boolean constant: -require(bool,string)(I_C(_msgSender()) == false,Just Wallet) (Smart_DeFi_NetWork.sol#34)

De.Fi

Issue ID	170
Severity	**Informational**
Status	**High**

Description Code	**function Point_BroadCast() external** {
	require(I_C(_msgSender()) == false, "Just Wallet");
	require(DX(_msgSender()), "Owner Not Exist");
	require(newMember >= 5 , " After 5 BeCome_Owner ");
	require(Waiting == false, " Processing "); Waiting = true ; ZB(); newMember = 0; Waiting = false ;}

Location	Smart_DeFi_NetWork.Point_BroadCast() (Smart_DeFi_NetWork.sol#35) compares to a boolean constant: -require(bool,string)(I_C(_msgSender()) == false,Just Wallet) (Smart_DeFi_NetWork.sol#35)

De.Fi

Issue ID	170
Severity	**Informational**
Status	**High**

Description Code	**function Point_BroadCast() external** {
	require(I_C(_msgSender()) == false, "Just Wallet");
	require(DX(_msgSender()), "Owner Not Exist");
	require(newMember >= 5 , " After 5 BeCome_Owner ");
	require(Waiting == false, " Processing "); Waiting = true ; ZB(); newMember = 0; Waiting = false ;}

Location	Smart_DeFi_NetWork.Point_BroadCast() (Smart_DeFi_NetWork.sol#35) compares to a boolean constant: -require(bool,string)(Waiting == false, Processing) (Smart_DeFi_NetWork.sol#35)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	**function ZB**() **private** { **address** ZC; **address** ZD; **for** (**uint256** k = 0; k < newMember; k++) { ZC = KW[KW[JJ[k]].UP].UP; ZD = KW[JJ[k]].UP; **if** (ZE(ZD) == true) { JL[DJ] = ZD; DJ++;} **while** (ZC != **address**(0)) { **if** (KW[ZD].YY == false) { KW[ZC].LT++; KW[ZC].AL++;} **else** { KW[ZC].RT++; KW[ZC].AR++;} **if** (ZE(ZC) == true) {JL[DJ] = ZC; DJ++;} ZD = ZC; ZC = KW[ZC].UP;}}}
Location	Smart_DeFi_NetWork.ZB() (Smart_DeFi_NetWork.sol#36) compares to a boolean constant: -ZE(ZD) == true (Smart_DeFi_NetWork.sol#36)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	**function ZB**() **private** { **address** ZC; **address** ZD; **for** (**uint256** k = 0; k < newMember; k++) { ZC = KW[KW[JJ[k]].UP].UP; ZD = KW[JJ[k]].UP; **if** (ZE(ZD) == true) { JL[DJ] = ZD; DJ++;} **while** (ZC != **address**(0)) { **if** (KW[ZD].YY == false) { KW[ZC].LT++; KW[ZC].AL++;} **else** { KW[ZC].RT++; KW[ZC].AR++;} **if** (ZE(ZC) == true) {JL[DJ] = ZC; DJ++;} ZD = ZC; ZC = KW[ZC].UP;}}}
Location	Smart_DeFi_NetWork.ZB() (Smart_DeFi_NetWork.sol#36) compares to a boolean constant: -ZE(ZC) == true (Smart_DeFi_NetWork.sol#36)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	**function ZB**() **private** { **address** ZC; **address** ZD; **for** (**uint256** k = 0; k < newMember; k++) { ZC = KW[KW[JJ[k]].UP].UP; ZD = KW[JJ[k]].UP; **if** (ZE(ZD) == true) { JL[DJ] = ZD; DJ++;} **while** (ZC != **address**(0)) { **if** (KW[ZD].YY == false) { KW[ZC].LT++; KW[ZC].AL++;} **else** { KW[ZC].RT++; KW[ZC].AR++;} **if** (ZE(ZC) == true) {JL[DJ] = ZC; DJ++;} ZD = ZC; ZC = KW[ZC].UP;}}}
Location	Smart_DeFi_NetWork.ZB() (Smart_DeFi_NetWork.sol#36) compares to a boolean constant: -KW[ZD].YY == false (Smart_DeFi_NetWork.sol#36)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	function _Change_Wallet(address X) external { require(X != address(0), "Dont Enter address 0"); require(changeSwitch == true , "Do After ChangeSwitch"); require(DX(_msgSender()),"You Are Not Exist"); require(I_C(X) == false, "New address can not be contract"); require(IRT(_msgSender())," Do After Reward"); if (Owner_All_Point(_msgSender()) > 1000) { require(ChCr[KW[_msgSender()].id] < 8, "Just 5 Times");} else{ require(ChCr[KW[_msgSender()].id] < 3 , "Just 3 Times");} require(!DX(X), "New Address Exist!"); require(DX(KW[_msgSender()].UP), "Your UpLine Not Exist"); require(((KW[_msgSender()].PO == address(0)) || (DX(KW[_msgSender()].PO) && (KW[_msgSender()].QO==address(0))) || (DX(KW[_msgSender()].PO) && DX(KW[_msgSender()].QO))) , "Your Directs Not Imported!"); require(Waiting == false, "Processing"); Waiting = true; Node memory A = KW[_msgSender()]; VV[A.id] = X; Node memory B = KW[A.PO]; B.UP = X; KW[A.PO] = B; Node memory C = KW[A.QO]; C.UP = X; KW[A.QO] = C; Node memory U = KW[A.UP]; if (A.YY == false) {U.PO = X ;} else {U.QO = X;} KW[A.UP] = U; KW[X] = A; ChCr[KW[X].id]++; ChCr[KW[_msgSender()].id]++; delete KW[_msgSender()]; Waiting = false;}
Location	Smart_DeFi_NetWork._Change_Wallet(address) (Smart_DeFi_NetWork.sol#37) compares to a boolean constant: -require(bool,string)(Waiting == false,Processing) (Smart_DeFi_NetWork.sol#37)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	function _Change_Wallet(address X) external { require(X != address(0), "Dont Enter address 0"); require(changeSwitch == true , "Do After ChangeSwitch"); require(DX(_msgSender()),"You Are Not Exist"); require(I_C(X) == false, "New address can not be contract"); require(IRT(_msgSender())," Do After Reward"); if (Owner_All_Point(_msgSender()) > 1000) { require(ChCr[KW[_msgSender()].id] < 8, "Just 5 Times");} else{ require(ChCr[KW[_msgSender()].id] < 3 , "Just 3 Times");} require(!DX(X), "New Address Exist!"); require(DX(KW[_msgSender()].UP), "Your UpLine Not Exist"); require(((KW[_msgSender()].PO == address(0)) || (DX(KW[_msgSender()].PO) && (KW[_msgSender()].QO==address(0))) || (DX(KW[_msgSender()].PO) && DX(KW[_msgSender()].QO))) , "Your Directs Not Imported!"); require(Waiting == false, "Processing"); Waiting = true; Node memory A = KW[_msgSender()]; VV[A.id] = X; Node memory B = KW[A.PO]; B.UP = X; KW[A.PO] = B; Node memory C = KW[A.QO]; C.UP = X; KW[A.QO] = C; Node memory U = KW[A.UP]; if (A.YY == false) {U.PO = X ;} else {U.QO = X;} KW[A.UP] = U; KW[X] = A; ChCr[KW[X].id]++; ChCr[KW[_msgSender()].id]++; delete KW[_msgSender()]; Waiting = false;}
Location	Smart_DeFi_NetWork._Change_Wallet(address) (Smart_DeFi_NetWork.sol#37) compares to a boolean constant: -require(bool,string)(changeSwitch == true,Do After ChangeSwitch) (Smart_DeFi_NetWork.sol#37)

De.Fi

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function** _Change_Wallet(**address** X) **external** { require(X != **address**(0), "Dont Enter address 0"); require(changeSwitch == true , "Do After ChangeSwitch"); require(DX(_msgSender()),"You Are Not Exist"); require(I_C(X) == false, "New address can not be contract"); require(IRT(_msgSender())," Do After Reward"); **if** (Owner_All_Point(_msgSender()) > 1000) { require(ChCr[KW[_msgSender()].id] < 8, "Just 5 Times");} **else**{ require(ChCr[KW[_msgSender()].id] < 3 , "Just 3 Times");} require(!DX(X), "New Address Exist!"); require(DX(KW[_msgSender()].UP), "Your UpLine Not Exist"); require(((KW[_msgSender()].PO == **address**(0)) || (DX(KW[_msgSender()].PO) && (KW[_msgSender()].QO==**address**(0))) || (DX(KW[_msgSender()].PO) && DX(KW[_msgSender()].QO))) , "Your Directs Not Imported!"); require(Waiting == false, "Processing"); Waiting = true; Node **memory** A = KW[_msgSender()]; VV[A.id] = X; Node **memory** B = KW[A.PO]; B.UP = X; KW[A.PO] = B; Node **memory** C = KW[A.QO]; C.UP = X; KW[A.QO] = C; Node **memory** U = KW[A.UP]; **if** (A.YY == false) {U.PO = X ;} **else** {U.QO = X;} KW[A.UP] = U; KW[X] = A; ChCr[KW[X].id]++; ChCr[KW[_msgSender()].id]++; **delete** KW[_msgSender()]; Waiting = false;}
Location	Smart_DeFi_NetWork._Change_Wallet(address) (Smart_DeFi_NetWork.sol#37) compares to a boolean constant: -A.YY == false (Smart_DeFi_NetWork.sol#37)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	function _Change_Wallet(address X) external { require(X != address(0), "Dont Enter address 0"); require(changeSwitch == true , "Do After ChangeSwitch"); require(DX(_msgSender()),"You Are Not Exist"); require(I_C(X) == false, "New address can not be contract"); require(IRT(_msgSender())," Do After Reward"); if (Owner_All_Point(_msgSender()) > 1000) { require(ChCr[KW[_msgSender()].id] < 8, "Just 5 Times");} else{ require(ChCr[KW[_msgSender()].id] < 3 , "Just 3 Times");} require(!DX(X), "New Address Exist!"); require(DX(KW[_msgSender()].UP), "Your UpLine Not Exist"); require(((KW[_msgSender()].PO == address(0)) || (DX(KW[_msgSender()].PO) && (KW[_msgSender()].QO==address(0))) || (DX(KW[_msgSender()].PO) && DX(KW[_msgSender()].QO))) , "Your Directs Not Imported!"); require(Waiting == false, "Processing"); Waiting = true; Node memory A = KW[_msgSender()]; VV[A.id] = X; Node memory B = KW[A.PO]; B.UP = X; KW[A.PO] = B; Node memory C = KW[A.QO]; C.UP = X; KW[A.QO] = C; Node memory U = KW[A.UP]; if (A.YY == false) {U.PO = X ;} else {U.QO = X;} KW[A.UP] = U; KW[X] = A; ChCr[KW[X].id]++; ChCr[KW[_msgSender()].id]++; delete KW[_msgSender()]; Waiting = false;}
Location	Smart_DeFi_NetWork._Change_Wallet(address) (Smart_DeFi_NetWork.sol#37) compares to a boolean constant: -require(bool,string)(I_C(X) == false,New address can not be contract) (Smart_DeFi_NetWork.sol#37)

De.Fi

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function _Emergency_Vote() external** { require(DX(_msgSender()), "Owner Not Exist"); require(Owner_All_Point(_msgSender()) > 0, "Just NetWorker"); require(voterExist(_msgSender()) == false, "You Did Vote Before"); voteTotal += Owner_All_Point(_msgSender()); VrL[VCr] = _msgSender(); VCr++;}
Location	Smart_DeFi_NetWork._Emergency_Vote() (Smart_DeFi_NetWork.sol#39) compares to a boolean constant: -require(bool,string)(voterExist(_msgSender()) == false,You Did Vote Before) (Smart_DeFi_NetWork.sol#39)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	**function ZH**(**address** A) **private view returns** (**uint32**) { **uint32** min = KW[A].LT <= KW[A].RT ? KW[A].LT : KW[A].RT; **if** (MaxPoint[A] == false){**if** (min > 5) {min = 5; }} **else** {**if** (min > 10) {min = 10;}} **return** min; }
Location	Smart_DeFi_NetWork.ZH(address) (Smart_DeFi_NetWork.sol#45) compares to a boolean constant: -MaxPoint[A] == false (Smart_DeFi_NetWork.sol#45)

De.Fi

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function DC2**(**address** Left_100, **address** Right_100) **private** { require(DX(_msgSender()),"Owner Not Exist"); require(MaxPoint[_msgSender()] == false, "You Did Max_Point"); require(Owner_Is_My_Line(KW[_msgSender()].PO,Left_100) == true, "Left_100 is not your line"); require(Owner_Is_My_Line(KW[_msgSender()].QO,Right_100) == true, "Right_100 is not your line"); require(Owner_All_Point(Left_100) >= 100, "Left_100 is not +100 point"); require(Owner_All_Point(Right_100) >= 100, "Right_100 is not +100 point"); MaxPoint[_msgSender()] = true; }
Location	Smart_DeFi_NetWork.DC2(address,address) (Smart_DeFi_NetWork.sol#75) compares to a boolean constant: -require(bool,string)(MaxPoint[_msgSender()] == false,You Did Max_Point) (Smart_DeFi_NetWork.sol#75)

De.Fi

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function DC2**(**address** Left_100, **address** Right_100) **private** { require(DX(_msgSender()),"Owner Not Exist"); require(MaxPoint[_msgSender()] == false, "You Did Max_Point"); require(Owner_Is_My_Line(KW[_msgSender()].PO,Left_100) == true, "Left_100 is not your line"); require(Owner_Is_My_Line(KW[_msgSender()].QO,Right_100) == true, "Right_100 is not your line"); require(Owner_All_Point(Left_100) >= 100, "Left_100 is not +100 point"); require(Owner_All_Point(Right_100) >= 100, "Right_100 is not +100 point"); MaxPoint[_msgSender()] = true; }
Location	Smart_DeFi_NetWork.DC2(address,address) (Smart_DeFi_NetWork.sol#75) compares to a boolean constant: -require(bool,string) (Owner_Is_My_Line(KW[_msgSender()].PO,Left_100) == true,Left_100 is not your line) (Smart_DeFi_NetWork.sol#75)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	**function DC2**(**address** Left_100, **address** Right_100) **private** { require(DX(_msgSender()),"Owner Not Exist"); require(MaxPoint[_msgSender()] == false, "You Did Max_Point"); require(Owner_Is_My_Line(KW[_msgSender()].PO,Left_100) == true, "Left_100 is not your line"); require(Owner_Is_My_Line(KW[_msgSender()].QO,Right_100) == true, "Right_100 is not your line"); require(Owner_All_Point(Left_100) >= 100, "Left_100 is not +100 point"); require(Owner_All_Point(Right_100) >= 100, "Right_100 is not +100 point"); MaxPoint[_msgSender()] = true; }
Location	Smart_DeFi_NetWork.DC2(address,address) (Smart_DeFi_NetWork.sol#75) compares to a boolean constant: -require(bool,string) (Owner_Is_My_Line(KW[_msgSender()].QO,Right_100) == true,Right_100 is not your line) (Smart_DeFi_NetWork.sol#75)

De.Fi

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function Agreement_Road_Map() external** { require(I_C(_msgSender()) == false, "Just Wallet"); require(Agreement_[_msgSender()] == false, "You Did Before "); Agreement_[_msgSender()] = true;}
Location	Smart_DeFi_NetWork.Agreement_Road_Map() (Smart_DeFi_NetWork.sol#77) compares to a boolean constant: -require(bool,string)(Agreement_[_msgSender()] == false,You Did Before) (Smart_DeFi_NetWork.sol#77)

De.Fi

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function Agreement_Road_Map() external** { require(I_C(_msgSender()) == false, "Just Wallet"); require(Agreement_[_msgSender()] == false, "You Did Before "); Agreement_[_msgSender()] = true;}
Location	Smart_DeFi_NetWork.Agreement_Road_Map() (Smart_DeFi_NetWork.sol#77) compares to a boolean constant: -require(bool,string)(I_C(_msgSender()) == false,Just Wallet) (Smart_DeFi_NetWork.sol#77)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	function _Switch_Change() external { require(_msgSender() == Agent, "Just Agent"); if (changeSwitch== false) {changeSwitch= true ;} else {changeSwitch= false ;}}
Location	Smart_DeFi_NetWork._Switch_Change() (Smart_DeFi_NetWork.sol#82) compares to a boolean constant: -changeSwitch == false (Smart_DeFi_NetWork.sol#82)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	function _Set_Smart_DeFi_Bank(address X) external { require(_msgSender() == Founder, "Just Founder"); require(Set_Bank == false, "Just 1 Time"); smart_Bank = X; Set_Bank = true; }
Location	Smart_DeFi_NetWork._Set_Smart_DeFi_Bank(address) (Smart_DeFi_NetWork.sol#87) compares to a boolean constant: -require(bool,string)(Set_Bank == false,Just 1 Time) (Smart_DeFi_NetWork.sol#87)

De.Fi

Issue ID	170
Severity	**Informational**
Status	**High**
Description Code	**function _Set_Smart_DeFi_Gift(address** X) **external** { require(_msgSender() == Founder, "Just Founder"); require(Set_Gift == false, "Just 1 Time"); smart_Gift = X; Set_Gift = true; }
Location	Smart_DeFi_NetWork._Set_Smart_DeFi_Gift(address) (Smart_DeFi_NetWork.sol#88) compares to a boolean constant: -require(bool,string)(Set_Gift == false,Just 1 Time) (Smart_DeFi_NetWork.sol#88)

De.Fi

Issue ID	170
Severity	Informational
Status	High
Description Code	function _Set_Import_Setup_Irreturnable(address R) external { require(_msgSender() == Founder, "Just Founder"); require(smart_Import == false, "Just 1 Times"); Smart_Import = R; smart_Import = true;}
Location	Smart_DeFi_NetWork._Set_Import_Setup_Irreturnable(address) (Smart_DeFi_NetWork.sol#89) compares to a boolean constant: -require(bool,string)(smart_Import == false,Just 1 Times) (Smart_DeFi_NetWork.sol#89)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** All_Owner_Number() **public view returns** (**uint64**) {**return** JK;}
Location	All_Owner_Number() should be declared external: - Smart_DeFi_NetWork.All_Owner_Number() (Smart_DeFi_NetWork.sol#50)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** All_Owner_Address(**uint32** start, **uint32** end) **public view returns** (**address**[] **memory**) { **uint32** index; **address**[] **memory** ret = **new** **address**[]((end - start) + 1); **for** (**uint32** i = start; i <= end; i++) { ret[index] = VV[i]; index++; } **return** ret; }
Location	All_Owner_Address(uint32,uint32) should be declared external: - Smart_DeFi_NetWork.All_Owner_Address(uint32,uint32) (Smart_DeFi_NetWork.sol#51)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function Last_Value_Point**() **public view returns** (**uint256**) {**return** ZM / 10**18; }
Location	Last_Value_Point() should be declared external: - Smart_DeFi_NetWork.Last_Value_Point() (Smart_DeFi_NetWork.sol#52)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	function Last_Reward_Writer() public view returns(address) {return JY;}
Location	Last_Reward_Writer() should be declared external: - Smart_DeFi_NetWork.Last_Reward_Writer() (Smart_DeFi_NetWork.sol#53)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Last_Total_Point() **public view returns** (**uint32**) {**return** ZL;}
Location	Last_Total_Point() should be declared external: - Smart_DeFi_NetWork.Last_Total_Point() (Smart_DeFi_NetWork.sol#54)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function Just_Contract_Balance**() **public view returns** (**uint256**) {**return** stableCoin.balanceOf(**address**(this)) / 10**18;}
Location	Just_Contract_Balance() should be declared external: - Smart_DeFi_NetWork.Just_Contract_Balance() (Smart_DeFi_NetWork.sol#56)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Owner_Big_Side(**address** R) **public view returns** (**uint32**) {**return** KW[R].AL >= KW[R].AR ? KW[R].AL : KW[R].AR; }
Location	Owner_Big_Side(address) should be declared external: - Smart_DeFi_NetWork.Owner_Big_Side(address) (Smart_DeFi_NetWork.sol#58)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Owner_Info_Global(**address** R) **public view returns** (Node **memory**) {**return** KW[R];}
Location	Owner_Info_Global(address) should be declared external: - Smart_DeFi_NetWork.Owner_Info_Global(address) (Smart_DeFi_NetWork.sol#60)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Owner_UpLine(**address** R) **public view returns** (**address**) {**return** KW[R].UP;}
Location	Owner_UpLine(address) should be declared external: - Smart_DeFi_NetWork.Owner_UpLine(address) (Smart_DeFi_NetWork.sol#61)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Owner_Directs(**address** R) **public view returns** (**address**, **address**) {**return** (KW[R].PO, KW[R].QO);}
Location	Owner_Directs(address) should be declared external: - Smart_DeFi_NetWork.Owner_Directs(address) (Smart_DeFi_NetWork.sol#62)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Owner_Left_Right_All (**address** R) **public view returns** (**uint32**, **uint32**) {**return** (KW[R].AL, KW[R].AR);}
Location	Owner_Left_Right_All(address) should be declared external: - Smart_DeFi_NetWork.Owner_Left_Right_All(address) (Smart_DeFi_NetWork.sol#63)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Owner_Left_Right_Save (**address** R) **public view returns** (**uint32**, **uint32**) {**return** (KW[R].LT, KW[R].RT);}
Location	Owner_Left_Right_Save(address) should be declared external: - Smart_DeFi_NetWork.Owner_Left_Right_Save(address) (Smart_DeFi_NetWork.sol#64)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Owner_All_Team (**address** R) **public view returns** (**uint32**) {**return** (KW[R].AL + KW[R].AR);}
Location	Owner_All_Team(address) should be declared external: - Smart_DeFi_NetWork.Owner_All_Team(address) (Smart_DeFi_NetWork.sol#65)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function Smart_History**() **public pure returns** (**address** Smart_Binance, **address** Smart_Binance_Pro, **address** Smart_Binance_Pro_2, **address** Smart_Binance_Pro_3){ **return** (0x5741da6D2937E5896e68B1604E25972a4834C701 , 0xFc46B09bf98858B08C5c5DEeb5c19E609FaBD398 , 0x8E60F00C14D5BB0B183a8e0a0e97737D254d906e , 0x8Aa1055188b407A58dF7d7737314d916A6F4ea24);}
Location	Smart_History() should be declared external: - Smart_DeFi_NetWork.Smart_History() (Smart_DeFi_NetWork.sol#66)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function Reward_Fee_Status**() **public view returns** (**uint256**) {**return** rewardFee;}
Location	Reward_Fee_Status() should be declared external: - Smart_DeFi_NetWork.Reward_Fee_Status() (Smart_DeFi_NetWork.sol#70)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function Reward_Counter_Status() public view returns (uint256) {return** RCr;}
Location	Reward_Counter_Status() should be declared external: - Smart_DeFi_NetWork.Reward_Counter_Status() (Smart_DeFi_NetWork.sol#71)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** _New_Owner_Status() **public view returns** (**uint256**) {**return** newMember;}
Location	_New_Owner_Status() should be declared external: - Smart_DeFi_NetWork._New_Owner_Status() (Smart_DeFi_NetWork.sol#72)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Owner_UpLines_All_Address(**address** R) **public view returns** (**address**[] **memory**) { **address**[] **memory** OUAL = **new address**[](JK); **uint32** OUAC; **address** _D_UpLine = KW[R].UP; **address** _D = R; **while** (_D != **address**(0)){ OUAL[OUAC] = _D_UpLine; OUAC++; _D = _D_UpLine; _D_UpLine = KW[_D_UpLine].UP;} **address**[] **memory** ret = **new address**[](OUAC); **for** (**uint32** i = 0 ; i < OUAC ; i++){ ret[i] = OUAL[i];} **return** ret;}
Location	Owner_UpLines_All_Address(address) should be declared external: - Smart_DeFi_NetWork.Owner_UpLines_All_Address(address) (Smart_DeFi_NetWork.sol#73)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Owner_Max_Point_Status(**address** Owner) **public view returns** (**bool**) {**return** MaxPoint[Owner];}
Location	Owner_Max_Point_Status(address) should be declared external: - Smart_DeFi_NetWork.Owner_Max_Point_Status(address) (Smart_DeFi_NetWork.sol#78)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** _Switch_Change_Status() **public view returns** (**bool**) { **return** changeSwitch ;}
Location	_Switch_Change_Status() should be declared external: - Smart_DeFi_NetWork._Switch_Change_Status() (Smart_DeFi_NetWork.sol#80)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	function _Write_Road_Map(string memory l) public {require(_msgSender() == Founder , " Just Founder "); Road_Map = l;}
Location	_Write_Road_Map(string) should be declared external: - Smart_DeFi_NetWork._Write_Road_Map(string) (Smart_DeFi_NetWork.sol#83)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** _Write_Founder_Message(**string memory** M) **public** {require(_msgSender() == Founder , " Just Founder "); Founder_Message = M;}
Location	_Write_Founder_Message(string) should be declared external: - Smart_DeFi_NetWork._Write_Founder_Message(string) (Smart_DeFi_NetWork.sol#84)

De.Fi

Issue ID	184
Severity	Optimization
Status	High
Description Code	**function** Smart_Road_Map_() **public view returns** (**string memory**) {**return** Road_Map;}
Location	Smart_Road_Map_() should be declared external: - Smart_DeFi_NetWork.Smart_Road_Map_() (Smart_DeFi_NetWork.sol#85)

De.Fi

Issue ID	177
Severity	Informational
Status	High
Description Code	**pragma solidity** >=0.4.22 <0.9.0;
Location	Pragma version>=0.4.22<0.9.0 (Smart_DeFi_NetWork.sol#2) is too complex

De.Fi

Issue ID	177
Severity	Informational
Status	High
Description Code	
Location	solc-0.8.22 is not recommended for deployment

De.Fi

Issue ID	173
Severity	**Informational**
Status	**High**
Description Code	**function sendValue**(**address payable** recipient, **uint256** amount) **internal** {require(**address**(this).balance >= amount, "Address: insufficient balance"); (**bool** success,) = recipient.call{value: amount}(""); require(success, "Address: unable to send value, recipient may have reverted");}
Location	Low level call in Address.sendValue(address,uint256) (Smart_DeFi_NetWork.sol#9): - (success) = recipient.call{value: amount}() (Smart_DeFi_NetWork.sol#9)

De.Fi

Issue ID	173
Severity	Informational
Status	High
Description Code	**function** _functionCallWithValue(**address** target, **bytes memory** data, **uint256** weiValue, **string memory** errorMessage) **private returns** (**bytes memory**) {require(isContract(target), "Address: call to non-contract"); (**bool** success, **bytes memory** returndata) = target.call{value: weiValue}(data); **if** (success) {**return** returndata; } **else** { **if** (returndata.length > 0) { **assembly** {**let** returndata_size := mload(returndata) **revert**(add(32, returndata), returndata_size)}} **else** { **revert**(errorMessage);}}}}
Location	Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Smart_DeFi_NetWork.sol#14): - (success,returndata) = target.call{value: weiValue}(data) (Smart_DeFi_NetWork.sol#14)

De.Fi

Issue ID	168
Severity	Low
Status	Medium
Description Code	function _Set_Smart_DeFi_Bank(address X) external { require(_msgSender() == Founder, "Just Founder"); require(Set_Bank == false, "Just 1 Time"); smart_Bank = X; Set_Bank = true; }
Location	Smart_DeFi_NetWork._Set_Smart_DeFi_Bank(address).X (Smart_DeFi_NetWork.sol#87) lacks a zero-check on : - smart_Bank = X (Smart_DeFi_NetWork.sol#87)

De.Fi

Issue ID	168
Severity	**Low**
Status	**Medium**
Description Code	**function _Set_Smart_DeFi_Gift(address** X) **external** { require(_msgSender() == Founder, "Just Founder"); require(Set_Gift == false, "Just 1 Time"); smart_Gift = X; Set_Gift = true; }
Location	Smart_DeFi_NetWork._Set_Smart_DeFi_Gift(address).X (Smart_DeFi_NetWork.sol#88) lacks a zero-check on : - smart_Gift = X (Smart_DeFi_NetWork.sol#88)

De.Fi

Issue ID	168
Severity	Low
Status	Medium
Description Code	function _Set_Import_Setup_Irreturnable(address R) external { require(_msgSender() == Founder, "Just Founder"); require(smart_Import == false, "Just 1 Times"); Smart_Import = R; smart_Import = true;}
Location	Smart_DeFi_NetWork._Set_Import_Setup_Irreturnable(address).R (Smart_DeFi_NetWork.sol#89) lacks a zero-check on : - Smart_Import = R (Smart_DeFi_NetWork.sol#89)

De.Fi

Issue ID	152
Severity	**Informational**
Status	**High**

Description Code	**function _Set_Stable_Coin(uint8** R) **external** {
	require(_msgSender() == Agent, "Just Agent");
	require(R >= 0 && R < 6, "Just 0,1,2,3,4,5"); **address**[6] **memory** C = [
	0x55d398326f99059fF775485246999027B3197955,
	0x8AC76a51cc950d9822D68b83fE1Ad97B32Cd580d,
	0x1AF3F329e8BE154074D8769D1FFa4eE058B1DBc3,
	0xc5f0f7b66764F6ec8C8Dff7BA683102295E16409,
	0x40af3827F39D0EAcBF4A168f8D4ee67c121D11c9,
	0x392004BEe213F1FF580C867359C246924f21E6Ad];
	stableCoin = IERC20(C[R]);}

Location	Smart_DeFi_NetWork._Set_Stable_Coin(uint8) (Smart_DeFi_NetWork.sol#86) contains a tautology or contradiction: - require(bool,string)(R >= 0 && R < 6,Just 0,1,2,3,4,5) (Smart_DeFi_NetWork.sol#86)

De.Fi

Issue ID	158
Severity	Informational
Status	Medium
Description Code	function BeCome_Owner(address Up) external {DC(Up);} function DC(address Up) private { require(I_C(_msgSender()) == false, "Just Wallet"); require(Up != address(0), "Dont Enter address 0"); require(KW[Up].XI != 2, " Upline Has 2 Directs "); require(_msgSender() != Up, " Dont Enter Your Address "); require(!DX(_msgSender()), " You Are An Owner "); require(DX(Up), " Upline Not Exist "); require(Agreement_[_msgSender()] == true, " Write Agreement"); require(Owner_Is_My_Line(Founder_Wallet , Up) == true, "UpLines Are Not Imported"); require(Waiting == false, " Processing "); Waiting = true; stableCoin.safeTransferFrom(_msgSender(), address(this), 100 * 10**18); IERC20(stableCoin).transfer(smart_Gift, 5 * 10**18); VV[JK] = _msgSender(); JK++; Node memory owner = Node({ id: JK, AL: 0, AR: 0, LT: 0, RT: 0, XI: 0 , YY: KW[Up].XI == 0 ? false : true, UP: Up, PO: address(0), QO: address(0)}); KW[_msgSender()] = owner; JJ[newMember] = _msgSender(); DZ++;newMember++; if (KW[Up].XI == 0) {KW[Up].LT+=1; KW[Up].AL+=1; KW[Up].PO = _msgSender();} else {KW[Up].RT+=1; KW[Up].AR+=1; KW[Up].QO = _msgSender();} KW[Up].XI+=1; Waiting = false;}
Location	Smart_DeFi_NetWork.DC(address) (Smart_DeFi_NetWork.sol#33) ignores return value by IERC20(stableCoin).transfer(smart_Gift,5 * 10 ** 18) (Smart_DeFi_NetWork.sol#33)

De.Fi

Issue ID	160
Severity	Informational
Status	Medium
Description Code	function Owner_Is_My_Line(address Up_Line, address Down_Line) public view returns (bool){ if (Up_Line == Down_Line) { return true;} else{ address E= KW[Down_Line].UP; bool temp; while (E != address(0)) { if (E== Up_Line){temp = true; break;} E= KW[E].UP; } if (temp){return true;} else {return false;} }}
Location	Smart_DeFi_NetWork.Owner_Is_My_Line(address,address).temp (Smart_DeFi_NetWork.sol#76) is a local variable never initialized

Issue ID	135
Severity	**Informational**
Status	**High**
Description Code	**contract Smart_DeFi_NetWork is Context**{**using SafeERC20 for IERC20**; **struct Node** { **uint64** id; **uint32** AL; **uint32** AR; **uint32** LT; **uint32** RT; **uint8** XI; **bool** YY; **address** UP; **address** PO; **address** QO;} **mapping**(**address** => Node) **internal** KW; **mapping**(**address** => **uint8**) **internal** EE; **mapping**(**uint64** => **address**) **internal** VV; **mapping**(**uint64** => **uint32**) **internal** QF; **mapping**(**uint64** => **uint32**) **internal** ChCr; **mapping**(**uint256** => **address**) **internal** JJ; **mapping**(**uint256** => **uint256**) **internal** VPL; **mapping**(**uint32** => **address**) **internal** JL; **mapping**(**uint32** => **address**) **internal** VrL; **mapping**(**address** => **bool**) **internal** MaxPoint; **mapping**(**address** => **bool**) **internal** Agreement_; **address internal** Founder; **address internal** JY; **address internal** smart_Bank; **address internal** smart_Gift; **address internal** Founder_Wallet; **address internal** Agent; **address internal** Smart_Import; IERC20 **internal** stableCoin; **uint64 internal** JK; **uint24 internal** DJ; **uint32 internal** ZL; **uint32 internal** voteTotal; **uint32 internal** VCr; **uint256 internal** time; **uint256 internal** RCr; **uint256 internal** ZM; **uint256 internal** DZ; **uint256 internal** LZ; **uint256 internal** rewardFee; **uint256 internal** newMember; **bool internal** Waiting; **bool internal** Set_Bank; **bool internal** Set_Gift; **bool internal** smart_Import; **bool internal** changeSwitch; **string internal** Road_Map; **string internal** Founder_Message;
Location	Smart_DeFi_NetWork.Waiting (Smart_DeFi_NetWork.sol#31) is written in both Waiting = true (Smart_DeFi_NetWork.sol#35) Waiting = false (Smart_DeFi_NetWork.sol#35)