BlockSAFU

# ADVANCE MANUAL
# SMART CONTRACT AUDIT


**Project:** From Hell

**Website: https://www.fromhell.info/**


**BlockSAFU Score:**

# 97

**Contract Address:**

**0x19d726cd37CA413adf46D9Ae555aB5A5643EdF20**

# DISCLAMER

BlockSAFU has completed this report to provide a summary of the Smart Contract functions, and any security, dependency, or cybersecurity vulnerabilities. This is often a constrained report on our discoveries based on our investigation and understanding of the current programming versions as of this report's date. To understand the full scope of our analysis, it is vital for you to at the date of this report. To understand the full scope of our analysis, you need to review the complete report. Although we have done our best in conducting our investigation and creating this report, it is vital to note that you should not depend on this report and cannot make any claim against BlockSAFU or its Subsidiaries and Team members on the premise of what has or has not been included in the report. Please remember to conduct your independent examinations before making any investment choices. We do not provide investment advice or in any way claim to determine if the project will be successful or not.

By perusing this report or any portion of it, you concur to the terms of this disclaimer. In the unlikely situation where you do not concur to the terms, you should immediately terminate reading this report, and erase and discard any duplicates of this report downloaded and/or printed by you. This report is given for data purposes as it were and on a non-reliance premise and does not constitute speculation counsel. No one should have any right to depend on the report or its substance, and BlockSAFU and its members (including holding companies, shareholders, backups, representatives, chiefs, officers, and other agents) BlockSAFU and its subsidiaries owe no obligation of care towards you or any other person, nor does BlockSAFU make any guarantee or representation to any individual on the precision or completeness of the report.

ABOUT THE AUDITOR:

BlockSAFU (BSAFU) is an Anti-Scam Token Utility that reviews Smart Contracts and Token information to Identify Rug Pull and Honey Pot scamming activity. BlockSAFUs Development Team consists of several Smart Contract creators, Auditors Developers, and Blockchain experts. BlockSAFU provides solutions, prevents, and hunts down scammers. BSAFU is a utility token with features Audit, KYC, Token Generators, and Bounty Scammers. It will enrich the crypto ecosystem.

## OVERVIEW

**BlockSAFU was commissioned by From Hell to complete a Smart Contract audit. The objective of the Audit is to achieve the following:**

- Review the Project and experience and Development team
- Ensure that the Smart Contract functions are necessary and operate as intended.
- Identify any vulnerabilities in the Smart Contract code.

DISCLAIMER: This Audit is intended to inform about token Contract Risks, the result does not imply an endorsement or provide financial advice in any way, all investments are made at your own risk. (https://blocksafu.com/)

# SMART CONTRACT REVIEW

| | |
|---|---|
| Token Name | **From Hell Universe** |
| Token Symbol | **HELL** |
| Token Decimal | 18 |
| Total Supply | 500,000,000 **DARK** |
| Contract Address | 0x19d726cd37CA413adf46D9Ae555aB5A5643EdF20 |
| Deployer Address | 0xC2519999D250848c8c3691Dd6D6335c8c9474685 |
| Owner Address | 0xC2519999D250848c8c3691Dd6D6335c8c9474685 |
| Tax Fees Buy | 0% |
| Tax Fees Sell | 0% |
| Gas Used for Buy | *will be updated after the DEX listing* |
| Gas Used for Sell | *will be updated after the DEX listing* |
| Contract Created | Sep-02-2022 12:25:24 PM +UTC |
| Initial Liquidity | *will be updated after the DEX listing* |
| Liquidity Status | Locked |
| Unlocked Date | *will be updated after the DEX listing* |
| Verified CA | Yes |
| Compiler | v0.8.4+commit.c7e474f2 |
| Optimization | Yes with 5000 runs |
| Sol License | MIT License |
| Top 5 Holders | *will be updated after the DEX listing* |
| Other | default evmVersion |

# TAX

| | | | |
|---|---|---|---|
| **BUY** | 0% | **SELL** | 0% |

## OVERVIEW

Mint Function

- No mint functions.

Fees

- Buy 0% (owner can’t set fees).
- Sell 0% (owner can’t set fees).

Tx Amount

- Owner cannot set a max tx amount.

Transfer Pausable

- Owner cannot pause.

Blacklist

- Owner cannot set a blacklist.

Ownership

- Owner cannot take back ownership.

Proxy

- This contract has no proxy.

Anti Whale

- Owner cannot limit the number of wallet holdings.

Trading Cooldown

- Owner cannot set the selling time interval.

## Token Holder

| Rank | Address | Quantity | Percentage | Analytics |
|---|---|---|---|---|
| 1 | 0xc2519999d250848c8c3691dd6d6335c8c9474685 | 500,000,000 | 100.0000% | |

[ Download CSV Export ]

## Team Review

The From Hell Universe team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 1 people in their telegram group (count in audit date).

## Official Website And Social Media

Website: https://www.fromhell.info/

Telegram Group: https://t.me/Fromhellann

Twitter: https://twitter.com/N_erdGames

## MANUAL CODE REVIEW

### Minor-risk

1 minor-risk code issue found

Could be fixed, and will not bring problems.

1. The return value of an external transfer/transferFrom return value is checked. Recommendation: use SafeERC20, or ensure that the transfer/transferFrom return value is checked

```
function transferFrom(
      address sender,
      address recipient,
      uint256 amount
   ) external returns (bool);
```

### Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

### High-Risk

0 high-risk code issues found

Must be fixed, and will bring problem.

### Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problem.

# EXTRA NOTES SMART CONTRACT

## 1. IERC20

```
interface IERC20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns
(uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to
`recipient`.
     *
     * Returns a boolean value indicating whether the operation
succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address recipient, uint256 amount) external
returns (bool);

    /**
     * @dev Returns the remaining number of tokens that `spender`
will be
     * allowed to spend on behalf of `owner` through
{transferFrom}. This is
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are
called.
     */
    function allowance(address owner, address spender) external
view returns (uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the
```

```
caller's tokens.
     *
     * Returns a boolean value indicating whether the operation
succeeded.
     *
     * IMPORTANT: 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
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount) external
returns (bool);

    /**
     * @dev Moves `amount` tokens from `sender` to `recipient`
using the
     * allowance mechanism. `amount` is then deducted from the
caller's
     * allowance.
     *
     * Returns a boolean value indicating whether the operation
succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);

    /**
     * @dev Emitted when `value` tokens are moved from one account
```

```
(`from`) to
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address indexed to,
uint256 value);

    /**
     * @dev Emitted when the allowance of a `spender` for an
`owner` is set by
     * a call to {approve}. `value` is the new allowance.
     */
    event Approval(address indexed owner, address indexed spender,
uint256 value);
}
```

ERC20 Normal Base Template

### 3. Safemath

```
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned integers, with an
overflow flag.
     *
     * _Available since v3.4._
     */
    function tryAdd(uint256 a, uint256 b) internal pure returns
(bool, uint256) {
        unchecked {
            uint256 c = a + b;
            if (c < a) return (false, 0);
            return (true, c);
        }
    }

    /**
     * @dev Returns the substraction of two unsigned integers,
with an overflow flag.
     *
     * _Available since v3.4._
     */
    function trySub(uint256 a, uint256 b) internal pure returns
(bool, uint256) {
        unchecked {
            if (b > a) return (false, 0);
            return (true, a - b);
        }
    }

    /**
     * @dev Returns the multiplication of two unsigned integers,
with an overflow flag.
     *
     * _Available since v3.4._
     */
    function tryMul(uint256 a, uint256 b) internal pure returns
(bool, uint256) {
        unchecked {
            // 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-contracts/pull/522
            if (a == 0) return (true, 0);
            uint256 c = a * b;
            if (c / a != b) return (false, 0);
            return (true, c);
        }
    }

    /**
     * @dev Returns the division of two unsigned integers, with a
division by zero flag.
     *
     * _Available since v3.4._
     */
    function tryDiv(uint256 a, uint256 b) internal pure returns
(bool, uint256) {
        unchecked {
            if (b == 0) return (false, 0);
            return (true, a / b);
        }
    }

    /**
     * @dev Returns the remainder of dividing two unsigned
integers, with a division by zero flag.
     *
     * _Available since v3.4._
     */
    function tryMod(uint256 a, uint256 b) internal pure returns
(bool, uint256) {
        unchecked {
            if (b == 0) return (false, 0);
            return (true, a % b);
        }
    }

    /**
     * @dev Returns the addition of two unsigned integers,
reverting on
     * overflow.
```

```
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     *
     * - Addition cannot overflow.
     */
    function add(uint256 a, uint256 b) internal pure returns
(uint256) {
        return a + b;
    }

    /**
     * @dev Returns the subtraction of two unsigned integers,
reverting on
     * overflow (when the result is negative).
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(uint256 a, uint256 b) internal pure returns
(uint256) {
        return a - b;
    }

    /**
     * @dev Returns the multiplication of two unsigned integers,
reverting on
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     *
     * - Multiplication cannot overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns
(uint256) {
        return a * b;
```

```
    }

    /**
     * @dev Returns the integer division of two unsigned integers,
reverting on
     * division by zero. The result is rounded towards zero.
     *
     * Counterpart to Solidity's `/` operator.
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(uint256 a, uint256 b) internal pure returns
(uint256) {
        return a / b;
    }

    /**
     * @dev Returns the remainder of dividing two unsigned
integers. (unsigned integer modulo),
     * reverting when dividing by zero.
     *
     * Counterpart to Solidity's `%` operator. This function uses
a `revert`
     * opcode (which leaves remaining gas untouched) while
Solidity uses an
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(uint256 a, uint256 b) internal pure returns
(uint256) {
        return a % b;
    }

    /**
     * @dev Returns the subtraction of two unsigned integers,
reverting with custom message on
     * overflow (when the result is negative).
```

```
     *
     * CAUTION: This function is deprecated because it requires
allocating memory for the error
     * message unnecessarily. For custom revert reasons use
{trySub}.
     *
     * Counterpart to Solidity's `-` operator.
     *
     * Requirements:
     *
     * - Subtraction cannot overflow.
     */
    function sub(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        unchecked {
            require(b <= a, errorMessage);
            return a - b;
        }
    }

    /**
     * @dev Returns the integer division of two unsigned integers,
reverting with custom message on
     * division by zero. The result is rounded towards zero.
     *
     * Counterpart to Solidity's `/` operator. Note: this function
uses a
     * `revert` opcode (which leaves remaining gas untouched)
while Solidity
     * uses an invalid opcode to revert (consuming all remaining
gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function div(
        uint256 a,
        uint256 b,
```

```
        string memory errorMessage
    ) internal pure returns (uint256) {
        unchecked {
            require(b > 0, errorMessage);
            return a / b;
        }
    }

    /**
     * @dev Returns the remainder of dividing two unsigned
integers. (unsigned integer modulo),
     * reverting with custom message when dividing by zero.
     *
     * CAUTION: This function is deprecated because it requires
allocating memory for the error
     * message unnecessarily. For custom revert reasons use
{tryMod}.
     *
     * Counterpart to Solidity's `%` operator. This function uses
a `revert`
     * opcode (which leaves remaining gas untouched) while
Solidity uses an
     * invalid opcode to revert (consuming all remaining gas).
     *
     * Requirements:
     *
     * - The divisor cannot be zero.
     */
    function mod(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        unchecked {
            require(b > 0, errorMessage);
            return a % b;
        }
    }
}
```

Standard Safemath

### 3. From Hell Universe Contract

```
contract StandardToken is IERC20, Ownable, BaseToken {
    using SafeMath for uint256;

    uint256 public constant VERSION = 1;

    mapping(address => uint256) private _balances;
    mapping(address => mapping(address => uint256)) private
_allowances;

    string private _name;
    string private _symbol;
    uint8 private _decimals;
    uint256 private _totalSupply;

    constructor(
        string memory name_,
        string memory symbol_,
        uint8 decimals_,
        uint256 totalSupply_,
        address serviceFeeReceiver_,
        uint256 serviceFee_
    ) payable {
        _name = name_;
        _symbol = symbol_;
        _decimals = decimals_;
        _mint(owner(), totalSupply_);

        emit TokenCreated(owner(), address(this),
TokenType.standard, VERSION);

        payable(serviceFeeReceiver_).transfer(serviceFee_);
    }

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

    /**
```

```
     * @dev Returns the symbol of the token, usually a shorter
version of the
     * name.
     */
    function symbol() public view virtual returns (string memory)
{
        return _symbol;
    }

    /**
     * @dev Returns the number of decimals used to get its user
representation.
     * For example, if `decimals` equals `2`, a balance of `505`
tokens should
     * be displayed to a user as `5,05` (`505 / 10 ** 2`).
     *
     * Tokens usually opt for a value of 18, imitating the
relationship between
     * Ether and Wei. This is the value {ERC20} uses, unless
{_setupDecimals} is
     * called.
     *
     * NOTE: This information is only used for _display_ purposes:
it in
     * no way affects any of the arithmetic of the contract,
including
     * {IERC20-balanceOf} and {IERC20-transfer}.
     */
    function decimals() public view virtual returns (uint8) {
        return _decimals;
    }

    /**
     * @dev See {IERC20-totalSupply}.
     */
    function totalSupply() public view virtual override returns
(uint256) {
        return _totalSupply;
    }

    /**
     * @dev See {IERC20-balanceOf}.
```

```
     */
    function balanceOf(address account)
        public
        view
        virtual
        override
        returns (uint256)
    {
        return _balances[account];
    }

    /**
     * @dev See {IERC20-transfer}.
     *
     * Requirements:
     *
     * - `recipient` cannot be the zero address.
     * - the caller must have a balance of at least `amount`.
     */
    function transfer(address recipient, uint256 amount)
        public
        virtual
        override
        returns (bool)
    {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }

    /**
     * @dev See {IERC20-allowance}.
     */
    function allowance(address owner, address spender)
        public
        view
        virtual
        override
        returns (uint256)
    {
        return _allowances[owner][spender];
    }
```

```
    /**
     * @dev See {IERC20-approve}.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function approve(address spender, uint256 amount)
        public
        virtual
        override
        returns (bool)
    {
        _approve(_msgSender(), spender, amount);
        return true;
    }

    /**
     * @dev See {IERC20-transferFrom}.
     *
     * Emits an {Approval} event indicating the updated allowance.
This is not
     * required by the EIP. See the note at the beginning of
{ERC20}.
     *
     * Requirements:
     *
     * - `sender` and `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `amount`.
     * - the caller must have allowance for ``sender``'s tokens of
at least
     * `amount`.
     */
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) public virtual override returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(
            sender,
            _msgSender(),
```

```
            _allowances[sender][_msgSender()].sub(
                amount,
                "ERC20: transfer amount exceeds allowance"
            )
        );
        return true;
    }

    /**
     * @dev Atomically increases the allowance granted to
`spender` by the caller.
     *
     * This is an alternative to {approve} that can be used as a
mitigation for
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    function increaseAllowance(address spender, uint256
addedValue)
        public
        virtual
        returns (bool)
    {
        _approve(
            _msgSender(),
            spender,
            _allowances[_msgSender()][spender].add(addedValue)
        );
        return true;
    }

    /**
     * @dev Atomically decreases the allowance granted to
`spender` by the caller.
     *
     * This is an alternative to {approve} that can be used as a
mitigation for
```

```
     * problems described in {IERC20-approve}.
     *
     * Emits an {Approval} event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the caller of at least
     * `subtractedValue`.
     */
    function decreaseAllowance(address spender, uint256
subtractedValue)
        public
        virtual
        returns (bool)
    {
        _approve(
            _msgSender(),
            spender,
            _allowances[_msgSender()][spender].sub(
                subtractedValue,
                "ERC20: decreased allowance below zero"
            )
        );
        return true;
    }

    /**
     * @dev Moves tokens `amount` from `sender` to `recipient`.
     *
     * This is internal function is equivalent to {transfer}, and
can be used to
     * e.g. implement automatic token fees, slashing mechanisms,
etc.
     *
     * Emits a {Transfer} event.
     *
     * Requirements:
     *
     * - `sender` cannot be the zero address.
     * - `recipient` cannot be the zero address.
     * - `sender` must have a balance of at least `amount`.
```

```
     */
    function _transfer(
        address sender,
        address recipient,
        uint256 amount
    ) internal virtual {
        require(sender != address(0), "ERC20: transfer from the
zero address");
        require(recipient != address(0), "ERC20: transfer to the
zero address");

        _beforeTokenTransfer(sender, recipient, amount);

        _balances[sender] = _balances[sender].sub(
            amount,
            "ERC20: transfer amount exceeds balance"
        );
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
    }

    /** @dev Creates `amount` tokens and assigns them to
`account`, increasing
     * the total supply.
     *
     * Emits a {Transfer} event with `from` set to the zero
address.
     *
     * Requirements:
     *
     * - `to` cannot be the zero address.
     */
    function _mint(address account, uint256 amount) internal
virtual {
        require(account != address(0), "ERC20: mint to the zero
address");

        _beforeTokenTransfer(address(0), account, amount);

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

```
    }

    /**
     * @dev Destroys `amount` tokens from `account`, reducing the
     * total supply.
     *
     * Emits a {Transfer} event with `to` set to the zero address.
     *
     * Requirements:
     *
     * - `account` cannot be the zero address.
     * - `account` must have at least `amount` tokens.
     */
    function _burn(address account, uint256 amount) internal
virtual {
        require(account != address(0), "ERC20: burn from the zero
address");

        _beforeTokenTransfer(account, address(0), amount);

        _balances[account] = _balances[account].sub(
            amount,
            "ERC20: burn amount exceeds balance"
        );
        _totalSupply = _totalSupply.sub(amount);
        emit Transfer(account, address(0), amount);
    }

    /**
     * @dev Sets `amount` as the allowance of `spender` over the
`owner` s tokens.
     *
     * This internal function is equivalent to `approve`, and can
be used to
     * e.g. set automatic allowances for certain subsystems, etc.
     *
     * Emits an {Approval} event.
     *
     * Requirements:
     *
     * - `owner` cannot be the zero address.
     * - `spender` cannot be the zero address.
```

```
     */
    function _approve(
        address owner,
        address spender,
        uint256 amount
    ) internal virtual {
        require(owner != address(0), "ERC20: approve from the zero
address");
        require(spender != address(0), "ERC20: approve to the zero
address");

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }

    /**
     * @dev Sets {decimals} to a value other than the default one
of 18.
     *
     * WARNING: This function should only be called from the
constructor. Most
     * applications that interact with token contracts will not
expect
     * {decimals} to ever change, and may work incorrectly if it
does.
     */
    function _setupDecimals(uint8 decimals_) internal virtual {
        _decimals = decimals_;
    }

    /**
     * @dev Hook that is called before any transfer of tokens.
This includes
     * minting and burning.
     *
     * Calling conditions:
     *
     * - when `from` and `to` are both non-zero, `amount` of
``from``'s tokens
     * will be to transferred to `to`.
     * - when `from` is zero, `amount` tokens will be minted for
`to`.
```

```
     * - when `to` is zero, `amount` of ``from``'s tokens will be
burned.
     * - `from` and `to` are never both zero.
     *
     * To learn more about hooks, head to
xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
     */
    function _beforeTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}
}
```

Contract From Hell Universe

## READ CONTRACT (ONLY NEED TO KNOW)

1. VERSION
1 uint256
(Shows contract version)

2. decimals
18 uint8
(Function for read decimals)

3. name
From Hell Universe string
(Function for read Token name))

4. Owner
0xc2519999d250848c8c3691dd6d6335c8c9474685 address
(Function for read owner address)

5. symbol
Hell string
(Function for read token symbol)

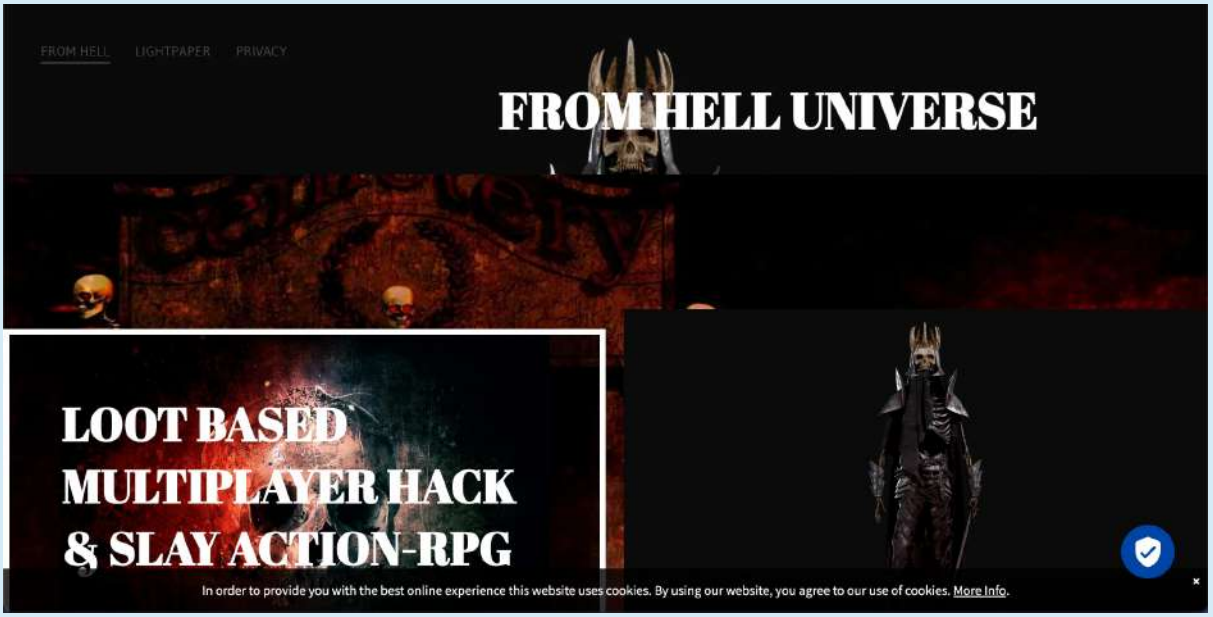
6. totalSupply
500000000000000000000000000 uint256
(Function for read total supply)

## WRITE CONTRACT

1. renounceOwnership
(Renouncing ownership will leave the contract without an owner, thereby removing any functionality that is only available to the owner)

2. transferOwnership
newOwner (address)
(Its function is to change the owner)

## WEBSITE REVIEW




- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By Sectigo SSL)**

### Web-Tech stack:

Domain .info (ionos) - Tracked by whois

| | |
|---|---|
| First Contentful Paint: | 781ms |
| Fully Loaded Time | 5.8s |
| Performance | 88% |
| Accessibility | 86% |
| Best Practices | 83% |
| SEO | 92% |

## RUG-PULL REVIEW

Based on the available information analyzed by us, we come to the following conclusions:

- Locked Liquidity (Locked by PinkSale)

  (*Will be updated after DEX listing*)

- TOP 5 Holder.

  (*Will be updated after DEX listing*)

- The Team KYC on PinkSale

## HONEYPOT REVIEW

- Ability to sell.
- The owner is not able to pause the contract.
- The owner can’t set fees

Note: Please check the disclaimer above. Note, the audit makes no statements or warranties on the business model, investment attractiveness, or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by the project owner.