# Stream Smart Business

Avaliações de segurança e conformidades do Contrato Inteligente

**StakHolders**

streamsmartbusiness.com

Dezembro de 2021

# INTRODUÇÃO

Esta auditoria tem resultado advindo das análises e testes do Smart Contract contra vetores comuns ou incomuns usados em ataques ao código, análises do ponto de vista de vulnerabilidades cibernéticas, e também se a estrutura do código segue as práticas recomendadas nos padrões da indústria.

Além disso, verificamos também se a intenção do projeto está de fato coerente com a lógica do código do contrato inteligente.

Os trabalhos se desenvolvem através de algumas metodologias, entre elas a revisão de todas linhas do código fonte por especialistas do setor.

As avaliações resultam em classificações que variam de críticas à aprovadas. Uma classificação é dada como crítica se uma potencial exploração do problema por invasores ou se a manifestação do bug puder causar sérias perdas financeiras ou interromper a execução do código.

Após a classificação crítica temos as classificações grandes, médias e baixa, com cada uma indicando o nível de gravidade.

As informativas são somente simples indicações acerca do código e contrato inteligente.

As aprovadas são todas aquelas que passaram em todas análises.

As indicações de **não encontrado** referem-se às situações não encontradas no contrato.

Os resultados produzidos por essa auditoria não devem eximir os responsáveis pelo projeto de também procurar outros revisores independentes, pois os Contratos Inteligentes e a blockchain são uma melhoria constante e muitos padrões e codificações estão em constante melhoria e atualização.

# VISÃO GERAL

O contrato possui somente um arquivo, o StakHolders.sol, que possui um código fonte com 844 linhas. Verificamos que todas as funções e variáveis foram usadas e comentadas de acordo com as documentações e padronizações da EPI ethereum network, bibliotecas aberta do OpenZeppelin e outras autoridades técnicas.

| | |
|---|---|
| **Nome do projeto** | StakHolders |
| **Código fonte** | https://bscscan.com/address/0x67ace44f45fd294ddcac5ebd11409512c47e6fd8#code |
| **Endereço do contrato** | 0x67ace44f45fd294ddcac5ebd11409512c47e6fd8 |
| **ByteCode for Ethereum Virtual Machine** | https://bscscan.com/bytecode-decompiler?a=0x77a8b4cc73c1486703af3190bbb154abd5618091 |
| **Taxas do contrato** | 0% compra e 12% venda |
| **Site** | stakholders.com.br |
| **Telegram** | t.me/Stakholders_Oficial |
| **Twitter** | twitter.com/stakholder |

## Descrição do projeto

O StakHolders é um projeto que objetiva criar uma comunidade integrada e unida para contribui positivamente para o meio BSC e para outros projetos de tokens. O projeto estaria, conforme declaração de seus fundadores, desenvolvendo plataformas e sistemas que ajudariam no engajamento do público.

## Descrição do token

O contrato possui funções que distribui dividendos em dólar BSC para todos detentores dos tokens; possui mecanismos que protegem a liquidez contra tentativas de trades de alta frequência por bots ou de alta volatilizada causada por dumps; também possui mecanismos de auto-liquidez.

# FUNCIONAMENTO BÁSICO

O contrato da StakHolders possui 12% em taxas na venda e zero na compra.

Do total, 1% é acumulado para conversão em BUSD (dólar da BSC) para posterior distribuição a todos os detentores existentes. Nesse mecanismo de pagamento de dividendos algumas condições de tempo e de quantidade de tokens devem ser atendidas para serem distribuídos os pagamentos.

Do total das taxas, outros 5% são acumulado internamente até que uma quantidade suficiente de capital tenha sido acumulada para realizar uma aquisição de LP no contrato de liquidez da PancakeSwap.

Os outros 6% em taxas são pagos à carteira do marketing do projeto para financiar os custos de divulgação e manutenção.

# INJEÇÃO DE LIQUIDEZ (AQUISIÇÃO DE LP)

O mecanismo de aquisição LP pode ser acionado indiretamente por qualquer transação normal do token, e ela avalia o conjunto de condições que acionam o mecanismo. As principais condições disso são se o remetente da transação é diferente do endereço do par de liquidez do token e se o limite de acúmulo de tokens no contrato foi ultrapassado. Se essas condições forem satisfeitas, a função swapThreshold é chamada diretamente do contrato da StakHolders.

# FUNÇÕES PRIVILEGIADAS

O contrato possui funções que apenas podem ser modificadas pelo proprietário do contrato, proprietário este definido por onlyOwner, seguindo o padrão ownable do OpenZeppelin, ou por um administrador definido como endereço autorizado no mapeamento authorizations.

As funções privilegiadas abaixo definem isenção de taxas, limites e dividendos:

- setIsTxLimitExempt (armazenamento em tipo address 20 bytes e tipo lógico booleano 1 ou zero), define a isenção de limite de transação para uma conta;

- setIsFeeExempt (armazenamento em tipo address 20 bytes e tipo lógico booleano 1 ou zero), define a isenção de taxas na transação;

- setIsDividendExempt (armazenamento em tipo address 20 bytes e tipo lógico booleano 1 ou zero), define a isenção de recebimento de dividendos;

- setTxLimit (armazenamento do tipo inteiro e sem sinal de 20 bytes), define a o limite geral de transação;

As funções privilegiadas abaixo definem as taxas e o endereço recebedor de ganhos em BNB:

- setFeeReceivers (armazenamento do tipo address 20 bytes), define a os endereços recebedores dos ganhos em BNB
- setFees (armazenamento do tipo inteiro e sem sinal de 20 bytes), define as taxas aplicadas nas vendas

O proprietário também tem outros privilégios em algumas funções, como por exemplo *modificar os valores do gás (taxa de rede)* nas transações e *configurar as condições das distribuições dos dividendos*

**Opinião do auditor e equipe**: o proprietário ter os privilégios acima elencados não é deveras problemático para o contrato ou para a comunidade. Alguns privilégios em funções são necessários para um correto gerenciamento do contrato, principalmente na tomada de decisões sobre a alteração das taxas cobradas.

# RENÚNCIA DE PROPRIEDADE DO CONTRATO

Verificamos que o código fonte não possui quaisquer definições para uma possível renúncia de propriedade do contrato para retirar do proprietário os controles privilegiados sobre o contrato. Não existem no mercado quaisquer recomendações por especialistas para a renúncia do contrato, sendo tão somente um desejo e pedido dos investidores.

**Opinião do auditor e equipe**: perde-se qualquer possibilidade de interagir, melhorar e corrigir um contrato renunciado. Por este motivo acreditamos ser uma boa prática de gerenciamento o proprietário-criador ter possibilidades de gerenciar o contrato

# ATAQUE POR ESTOURO DE MEMORY (OVERFLOW E UNDERFLOW)

Não identificamos possibilidade de exploração em ataques de overflow ou underflow (estouro positivo ou negativo), ou seja, ataques na falha da execução do contrato por erros resultantes de falta de espaço na memória atribuída à uma string, seja em tipos inteiros ou em pontos flutuantes nas operações matemáticas. O código fonte faz o correto e recomendado uso das bibliotecas SafeMath para cálculos das operações básicas da matemática.

**Opinião do auditor e equipe**: Esta prática mostra ser uma boa prática de código, já que o uso de um padrão validado e bem escrito protege contra execuções incorretas do código, contra tentativas de manipulação de valores em strings, ou contras brechas possíveis de serem exploradas por invasores.

# CONTRATO DE DIVIDENDOS NÃO POSSUI PREVENÇÃO CONTRA PERDA DE VALORES

O contrato de dividendo usado pelo contrato principal para distribuição de recompensas em BUSD não possui prevenção contra perda de tokens nele depositados.

O seguinte trecho de código estabelece o depósito de tokens BUSD no contrato de dividendos:

```
    function deposit() external payable override onlyToken {
        uint256 balanceBefore = BUSD.balanceOf(address(this));
        address[] memory path = new address[](2);
        path[0] = WBNB;
        path[1] = address(BUSD);
        router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: msg.value}(
            0,
            path,
            address(this),
            block.timestamp
        );
        uint256 amount = BUSD.balanceOf(address(this)).sub(balanceBefore);
        totalDividends = totalDividends.add(amount);
        dividendsPerShare =
dividendsPerShare.add(dividendsPerShareAccuracyFactor.mul(amount).div(totalShares));
    }
```

Os valores depositados não são cumulativos, isto é, sempre que as condições mínimas de tempo e valores são atingidas, o gatilho para distribuir os BUSD são acionados e o saldo do contrato zerado (ou quase zerado).

**Opinião do auditor e equipe**: é sempre uma boa prática a prevenção contra perda de valores em smarts contracts. Desse modo, o contrato nunca deve ser um vetor para possíveis perdas de valores nele presos. Classificação de gravidade: baixa.

# FUNÇÕES CONSIDERADA RUG-PULL

Não há no contrato funções ou mecanismos que podem ser considerados rug-pull, como por exemplo:

- mecanismos que realiza destruição (destruction or destroy function) de tokens da carteira do holder
- esquemas de criação deliberada de tokens para o proprietário (mint function)
- função de autodestruição do contrato (selfdestruct function) para dificultar rastreamento das transações

Há mecanismo com a finalidade de trancar/pausar o contrato. A equipe técnica justificou como sendo necessária para as etapas da venda privada e pré-venda.

**Opinião do auditor e equipe**: Foi verificado que se seguiu uma boa prática do mercado ao mostrar mais transparência e confiança para os investidores.

# TESTE DO CONTRATO EM TESTNET

Foram realizados vários testes do contrato em redes de testes, quais sejam: Ropsten, e BSC TestNet. Em todos testes o código fonte funcionou corretamente e conforme as intenções dos proprietários dos projetos.

# CONTRATO ACUMULA BNB

O acúmulo de BNB no contrato ocorre quando as funções características de swap são chamadas junto ao endereço do par de liquidez (compra ou venda). Quando isso ocorre, há uma diferença entre o preço de entrada e o preço de saída nas transações, e nessa diferença acabam por restar alguns poucos BNB da transação. Esses BNB são ***intencionalmente*** depositados no endereço do contrato, conforme dispõe as seguintes linhas de códigos:

```
uint256 balanceBefore = address(this).balance
```

e

```
uint256 amountBNB = address(this).balance.sub(balanceBefore)
```

onde sub é a função de operações matemáticas de adições da biblioteca SafeMath, que sempre adiciona ao saldo do contrato os valores restados da transação.

**Opinião do auditor e equipe:** Foi verificado que foram implementadas possibilidades de resolver o problema.

# REENTRANCY EXPLOIT

Não encontramos possibilidades da realização de ataques de reentrada por invasores, já que o contrato não utiliza funções call.value(). Somente função transfer() são utilizadas para finalidades de transferências, função esta que limita o uso de gás em 2.300 wei, impedindo de chamar recursivamente a mesma função em um ataque de reentrada.

# ATAQUE DDOS COM LIMITE DE GÁS POR BLOCO

As operações do contrato que necessitam de maior processamento possuem seu limite de gás devidamente estabelecido. Do mesmo modo, as funções de fallback estão corretamente implementadas para evitar um ataque de negação de processamento (DDoS attack)

# OTIMIZAÇÕES NO CONTRATO

O código fonte trabalha corretamente com armazenamentos de inteiros sem sinal no tamanho de 256 bits, já que esse é o tamanho padrão usado pela Ethereum Virtual Machine para processamento.

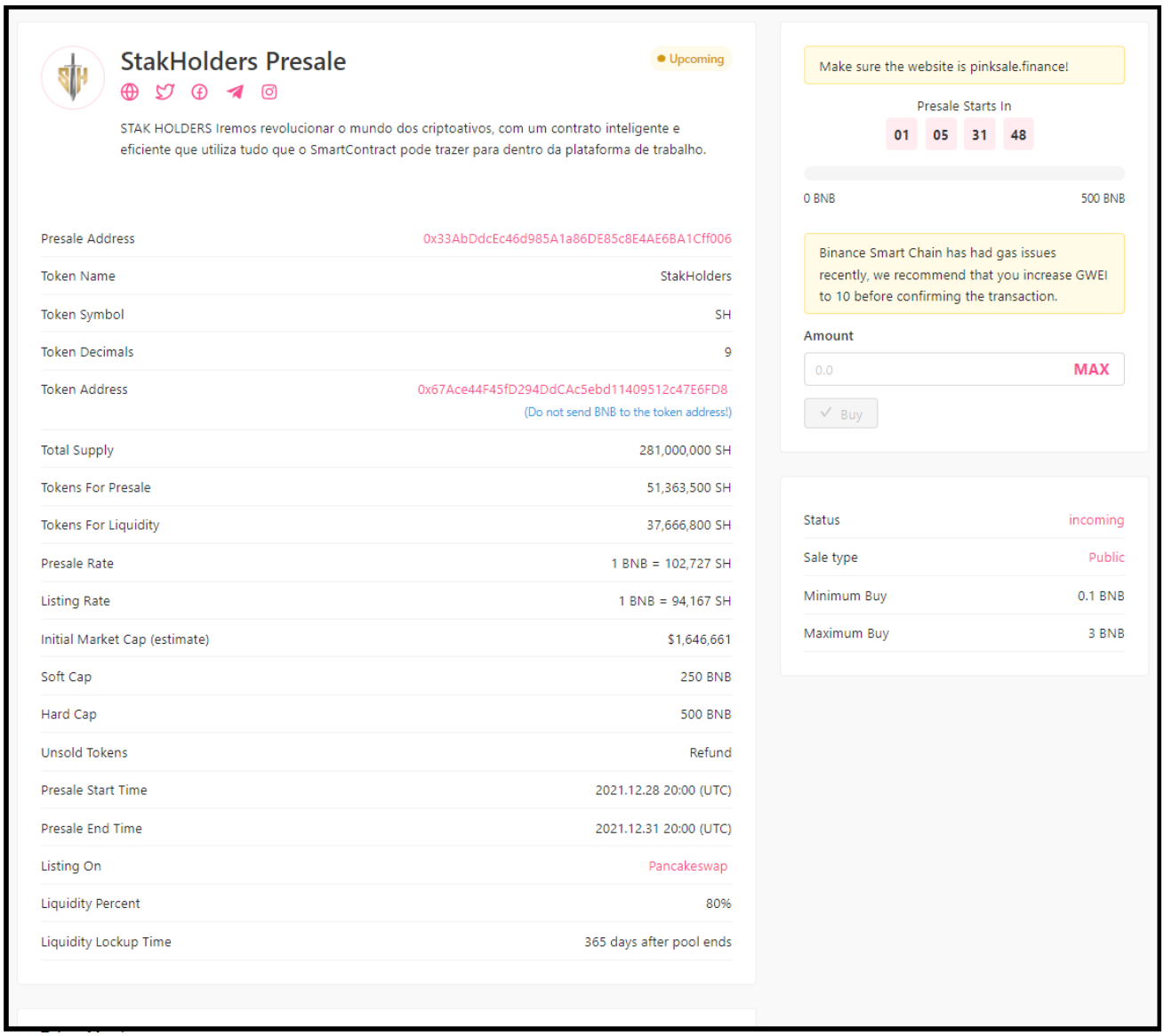
Verifica-se também que o contrato está com otimização para rodar em 2.000 execuções do seu bytecode, desse modo veremos gastos em gás mais otimizados para toda vida útil do mesmo.

# OUTRAS OBSERVAÇÕES

1. O contrato usa formatações require corretamente e em momentos ideais, se adequando às recomendações técnicas.
2. O contrato está em conformidade com as especificações mais comuns. As definições de token no contrato StakHolders.sol está em conforme o que dispõe a EIP-20 Token Standard.
3. O código fonte possui bons comentários para as partes mais complexas.
4. Os subcontratos utilizados (padrão ERC20, interface, router PancakeV2, dividendos) estão em conformidade com as recomendações técnicas.
5. No que tange a um código mais limpo e enxuto, verificamos um aviso do tipo informacional. Tal aviso não é uma advertência, mas sim, uma orientação informacional para um código mais leve, contribuindo para taxas muito menores na blockchain. Por outro lado, concordamos com a dificuldade em redução das linhas do código justamente pelos motivos do que o contrato pretende fazer (pagamentos de dividendos em BUSD para holders, BNB para os proprietários, taxas em transações...)

# BLOQUEIO DE LIQUIDEZ

O contrato está em fase de pré-venda na plataforma Dex PinkSale. No perfil de cadastro consta que a liquidez está com bloqueio de até 365 dias após a data de finalização da pré-venda. A finalização está prevista para dia 31 de dezembro de 2021.



# STATUS DAS CHECKAGENS NO CONTRATO

| | |
|---|---|
| Exatidão do código conforme as expectativas | APROVADO |
| Erros ao compilar | APROVADO |
| Dependência de timestamp | APROVADO |
| Testabilidade do código | APROVADO |
| Código limpo/enxuto | INFORMATIVAS |
| Ataque DDoS com Limite de gás por bloco | APROVADO |
| Ataque DDoS por falha nas chamadas das funções | APROVADO |
| Ataque DDoS por reversão no processamento das funções | APROVADO |
| Uso de funções ultrapassadas em Solidity | NÃO ENCONTRADO |
| Ataque de reentrada/reentrancy attack | APROVADO |
| Instruções SELFDESTRUCT mal documentada ou desprotegidas | NÃO ENCONTRADO |
| Não verificação do valor de retorno em funções Get | APROVADO |
| Estouro de overflow/underflow de inteiros | APROVADO |
| Presença de variáveis não usada | APROVADO |
| Redundância do código | APROVADO |
| Otimização de gás nas transações | INFORMATIVAS |
| Otimização da execução do contrato | APROVADO |
| Prevenção de perda de valores dentro do contrato | APROVADO |
| Violação do REQUIRE FUCTION | APROVADO |
| Erros de tipográfico | APROVADO |
| Retirada não protegida de BNB | APROVADO |
| Design lógico | INFORMATIVAS |
| Precisão aritmética | APROVADO |
| Segurança de funções fallback | APROVADO |
| Conformidade com os padrões EIPs | APROVADO |
| Implementação de bibliotecas padrão e seguras OpenZeppelin | APROVADO |

# CONCLUSÃO

A auditoria realizou análise manual linha por linha e revisão automatizada do smart contract. O contrato e código fonte foram analisados principalmente no ponto de vista de vulnerabilidades comuns de contratos inteligentes, exploits, hacks de manipulação, otimização e estrutura de código.

O contrato e código fonte possuem status de APROVADO nas análises e discursões da auditoria.

# AVISO LEGAL

A Stream Smart Business e sua equipe técnica fornece através deste relatório de auditoria apenas discussão e avaliação sobre projeto e contrato, com a única intenção de ajudar na melhoria da qualidade do código e a melhorar o nível de variação utilizado nas tecnologias de contratos inteligentes. Portanto, este relatório de auditoria não é um documento juridicamente vinculado.

Este relatório de auditoria técnica não é uma recomendação de compra ou venda, de aprovação ou desaprovação, ou mesmo uma resposta definitiva sobre os pontos aqui discutidos, mas expressa apenas a opinião técnica e analítica sobre o projeto StakHolders.

A reprodução deste material é livre e não há necessidade de solicitar permissão aos criadores. Porém, é necessária a citação da fonte.