Secure Auction Smart Contract - Subasta2.sol

Overview

This smart contract implements a secure and transparent auction system on Ethereum.
The auction duration is variable and defined by the owner at deployment time (in minutes).
If a valid bid is placed within the last 10 minutes, the auction is automatically extended by 10 minutes, up to a maximum equal to the original duration.

All important actions are recorded via events for greater transparency.
Only the owner can finalize the auction and refund non-winning bidders, deducting a 2% commission.
The contract is designed for academic and small-scale use.

Main Features

- **Variable auction duration:** The owner sets the duration (in minutes) when deploying the contract. If there are bids in the last 10 minutes, the auction is extended by 10 minutes (up to the maximum of the original duration).
- **Bidding:** Only valid bids are accepted (at least 5% higher than the current highest bid).
- **Partial withdrawal:** Participants can withdraw any excess deposit above their last valid bid during the auction.
- **Refunds:** After the auction ends, only the owner can refund non-winning bidders, deducting a 2% commission.
- **Emergency withdrawal:** The owner can recover all contract funds only if the auction was cancelled and no deposits remain.
- **Events:** All important actions emit events for transparency.

Key Variables

- **owner:** Address of the contract owner.
- **auctionEndTime:** Timestamp when the auction ends.
- **maxExtensionTime:** Maximum allowed extension (equal to the original duration).
- **extendedTime:** Total time the auction has been extended.

- **highestBidder:** Address of the current highest bidder.
- **highestBid:** Amount of the current highest bid.
- **bidHistory:** Array of all bids (address and amount).
- **deposits:** ETH deposits by address.
- **lastBid:** Last bid of each address.
- **bidIndex:** Index of each bidder in bidHistory.
- **hasBid:** Indicates if an address has already placed a bid.
- **lastBidTime:** Last time a user placed a bid.
- **ended:** True if the auction has ended.
- **fundsWithdrawn:** True if the owner has withdrawn the winning amount.
- **cancelled:** True if the auction was cancelled.

Main Functions

- **Constructor:** Initializes the auction with the duration defined by the owner (in minutes) and sets the deployer as the owner.
  Parameter: duration in minutes (e.g., 10080 for 7 days).
- **Bid:** Allows users (except the owner) to place bids. Each bid must be at least 5% higher than the current highest. Bids in the last 10 minutes extend the auction by 10 minutes (up to the maximum of the original duration). Emits the NewBid event.
  Parameter: msg.value (ETH sent with the bid).
- **Partial Withdrawal:** Allows bidders to withdraw any excess deposit above their last valid bid during the auction. Emits the PartialWithdrawal event.
- **Refund Non-Winners:** Only the owner can call this after the auction ends. Refunds non-winners, deducting a 2% commission. Emits
  the DepositWithdrawn and FeeTransferred events.
- **End Auction:** Allows the owner to manually end the auction before the scheduled time. Emits the AuctionEnded event.
- **Withdraw Winning Bid:** Allows the owner to withdraw the winning bid after the auction ends.

- **Cancel Auction:** Allows the owner to cancel the auction if there are no bids. Emits the AuctionCancelled event.
- **Withdraw Deposit After Cancellation:** Allows users to withdraw their deposit if the auction was cancelled. Emits the DepositWithdrawnOnCancel event.
- **Emergency Withdrawal:** Allows the owner to recover all ETH from the contract only if the auction was cancelled and no deposits remain. Emits the EmergencyWithdrawal event.
- **Get Bid Count:** Returns the number of bids placed in the auction.
- **Get Bid History (Paginated):** Returns a page of the bid history for pagination. Parameters: offset (start index), limit (number of bids to return).
- **Get Winner:** Returns the address of the highest bidder and the winning bid amount.

Events

- **NewBid:** Emitted when a new bid is placed.
- **AuctionEnded:** Emitted when the auction ends.
- **PartialWithdrawal:** Emitted when a user withdraws excess deposit.
- **DepositWithdrawn:** Emitted when a non-winner receives a refund.
- **AuctionCancelled:** Emitted when the auction is cancelled.
- **FeeTransferred:** Emitted when the owner receives the 2% commission.
- **DepositWithdrawnOnCancel:** Emitted when a user withdraws their deposit after cancellation.
- **EmergencyWithdrawal:** Emitted when the owner recovers all funds.

Security and Best Practices

- All critical functions use modifiers to restrict access and ensure the correct auction state.
- All ETH transfers use call and check for success.
- State changes are made before external calls to prevent reentrancy.

- All important actions emit events for transparency.
- The contract uses Solidity 0.8.x, which includes automatic overflow/underflow protection.
- The code is fully documented in English and uses NatSpec comments.
- Array lengths are stored in local variables before loops to optimize gas usage.

Limitations and Security Considerations

- **Gas Limit:** The withdrawDeposits function could hit the gas limit if there are too many bidders. For large-scale use, consider implementing individual withdrawals.
- **Emergency Withdrawal:** The owner can only recover all funds if the auction was cancelled and no deposits remain.
- **No Reentrancy Modifier:** The contract follows the checks-effects-interactions pattern, which is sufficient for this context.
- **Front-running:** The risk of front-running is inherent to public auctions on blockchain and is not specifically mitigated here.
- **Intended Use:** This contract is designed for academic and small-scale use. For production, thorough testing and a professional security audit are recommended.

Deployment and Verification

1. **Deployment:**
    - Use Remix IDE, Solidity 0.8.20, and deploy by entering the duration in minutes as a constructor parameter (for example, 10080 for 7 days).
2. **Verification:**
    - On Etherscan, select the correct compiler version and license, and paste the full contract code.
    - Enter the constructor parameter in ABI-encoded format (you can use ABI Hashex to obtain it).

Conclusion

This smart contract implements a secure and transparent auction system, following modern Solidity standards and best practices.
It is ready for academic evaluation and small-scale deployment. For production use, thorough testing and a professional security audit are strongly recommended.