```
import pandas as pd
import numpy as np

# 1. Load Data
data = pd.read_csv('BTC_2019_2023_4h.csv')
data['datetime'] = pd.to_datetime(data['datetime'])
data.set_index('datetime', inplace=True)

# 2. Indicator Calculations
def kama_er(close, period=20, fast_ema=2, slow_ema=30):
    change = abs(close - close.shift(period))
    volatility = abs(close - close.shift()).rolling(period).sum()
    er = change / volatility.replace(0, np.nan).fillna(0)
    sc = (2/(fast_ema+1) - 2/(slow_ema+1)) * er + 2/(slow_ema+1)
    sc = sc ** 2
    kama = close.copy()
    for i in range(period, len(close)):
        kama.iloc[i] = kama.iloc[i-1] + sc.iloc[i] * (close.iloc[i] - kama.iloc[i-1])
    return kama, er

data['KAMA'], data['ER'] = kama_er(data['close'])
data['RSI'] = data['close'].diff().apply(lambda x: x if x > 0 else 0).rolling(14).mean(
) / \
            data['close'].diff().abs().rolling(14).mean() * 100
data['VWAP'] = (data['close'] * data['volume']).cumsum() / data['volume'].cumsu
m()
data['Volume_MA'] = data['volume'].rolling(20).mean()
data['SMA_60'] = data['close'].rolling(20).mean()

# 3. Signal Generation
data['signals'] = 0
data['Trade_Type'] = 'Hold'  # 'Long', 'Short', 'Exit Long', 'Exit Short', or 'Hold'
data['Position'] = 0  # 1=Long, -1=Short, 0=Flat
```

```
data['StopLoss'] = np.nan
data['EntryPrice'] = np.nan
current_pos = 0
stop_loss = None

for i in range(60, len(data)):
    current_pos = data['Position'].iloc[i-1]
    current_price = data['close'].iloc[i]

    # ===== EXIT LOGIC =====
    if current_pos != 0:
        # Long Exit Conditions
        if current_pos == 1 and ((current_price <= stop_loss) or
                        (current_price < data['SMA_60'].iloc[i]) or (data['ER'].iloc[i] > 0
.75)):
            data.at[data.index[i], 'signals'] = -1  # Exit long
            data.at[data.index[i], 'Position'] = 0
            data.at[data.index[i], 'Trade_Type'] = 'Exit Long'
            stop_loss = None
            current_pos = 0

        # Short Exit Conditions
        elif current_pos == -1 and ((current_price >= stop_loss) or
                            (current_price > data['SMA_60'].iloc[i]) or (data['ER'].iloc[i] >
 0.75)):
            data.at[data.index[i], 'signals'] = 1  # Exit short
            data.at[data.index[i], 'Position'] = 0
            data.at[data.index[i], 'Trade_Type'] = 'Exit Short'
            stop_loss = None
            current_pos = 0

    # ===== ENTRY LOGIC ===== (Only if flat)
    if current_pos == 0:
```

```
    # Long Entry (Close > KAMA, ER > 0.4, RSI 40-70, Volume Spike)
    if (data['close'].iloc[i] > data['SMA_60'].iloc[i]) and (data['close'].iloc[i] > dat
a['KAMA'].iloc[i]) and\
        (data['ER'].iloc[i] > 0.4) and \
        (30 < data['RSI'].iloc[i] < 70) and \
        (data['volume'].iloc[i] > 1.1 * data['Volume_MA'].iloc[i]):
         data.at[data.index[i], 'signals'] = 1
         current_pos = 1
         data.at[data.index[i], 'Position'] = 1
         current_price = data['close'].iloc[i]
         data.at[data.index[i], 'EntryPrice'] = current_price
         stop_loss = current_price * 0.99 # 2% initial SL
         data.at[data.index[i], 'StopLoss'] = stop_loss
         data.at[data.index[i], 'Trade_Type'] = 'Long'

    # Short Entry (Close < KAMA, ER > 0.4, RSI 30-60, Volume Spike)
    elif (data['close'].iloc[i] < data['SMA_60'].iloc[i]) and (data['close'].iloc[i] < d
ata['KAMA'].iloc[i]) and\
         (data['ER'].iloc[i] > 0.4) and \
         (30 < data['RSI'].iloc[i] < 70) and \
         (data['volume'].iloc[i] > 1.1 * data['Volume_MA'].iloc[i]):
        data.at[data.index[i], 'signals'] = -1
        current_pos = -1
        data.at[data.index[i], 'Position'] = -1
        current_price = data['close'].iloc[i]
        data.at[data.index[i], 'EntryPrice'] = current_price
        stop_loss = current_price * 1.01  # 2% initial SL
        data.at[data.index[i], 'StopLoss'] = stop_loss
        data.at[data.index[i], 'Trade_Type'] = 'Short'

# 4. Post-Processing (Ensure proper signal sequence)
current_position = 0
for i in range(len(data)):
```

```
    if data['signals'].iloc[i] != 0:
        if current_position == 0:  # New position
            current_position = data['signals'].iloc[i]
        else:  # Exit position
            current_position = 0
    else:
        data.at[data.index[i], 'signals'] = 0

# 5. Save Results
output = data[['open', 'high', 'low', 'close', 'volume', 'signals', 'Trade_Type', 'Positio
n', 'EntryPrice', 'StopLoss']]
output.to_csv('BTC_ZeroNet_Signals_dhram2.csv')

print(f"Strategy Complete. Sample Signals:\n{output[output['signals'] != 0].head(
6)}")
```