MythX

Analysis 57efc1ec-42f0-42f4-af83-2781d1f2d0a8 MythX

Started	Fri Jun 16 2023 15:28:52 GMT+0000 (Coordinated Universal Time)
Finished	Fri Jun 16 2023 15:28:59 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Remythx
Main Source File	Contracts/DexillaExchangeV4.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	2	17

ISSUES

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
211 function average(uint256 a, uint256 b) internal pure returns (uint256) {
212 // (a + b) / 2 can overflow.
213 return (a & b) + (a ^ b) / 2;
214 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
211 function average(uint256 a, uint256 b) internal pure returns (uint256) {
212 // (a + b) / 2 can overflow.
213 return (a & b) + (a ^ b) / 2;
214 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
222 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
223 // (a + b - 1) / b can overflow on addition, so we distribute.
224 return a == 0 ? 0 : (a - 1) / b + 1;
225 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
222 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
223 // (a + b - 1) / b can overflow on addition, so we distribute.
224 return a == 0 ? 0 : (a - 1) / b + 1;
225 }
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
222 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
223 // (a + b - 1) / b can overflow on addition, so we distribute.
224 return a == 0 ? 0 : (a - 1) / b + 1;
225 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
245 // Handle non-overflow cases, 256 by 256 division.
246 if (prod1 == 0) {
247 return prod0 / denominator;
248 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
270
271 // Does not overflow because the denominator cannot be zero at this stage in the function.
272 uint256 twos = denominator & (~denominator + 1);
273 assembly {
274 // Divide denominator by twos.
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
283
284 // Shift in bits from prod1 into prod0.
285 prod0 |= prod1 * twos;
286
287 // Invert denominator mod 2^256. Now that denominator is an odd number, it has an inverse modulo 2^256 such
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
288 // that denominator * inv = 1 mod 2^256. Compute the inverse by starting with a seed that is correct for
289 // four bits. That is, denominator * inv = 1 mod 2^4.
290 uint256 inverse = (3 * denominator) ^ 2;
291
292 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
292 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
293 // in modular arithmetic, doubling the correct bits in each step.
294 inverse *= 2 - denominator * inverse; // inverse mod 2^8
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
292 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
293 // in modular arithmetic, doubling the correct bits in each step.
294 inverse *= 2 - denominator * inverse; // inverse mod 2^8
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
292 // Use the Newton-Raphson iteration to improve the precision. Thanks to Hensel's lifting lemma, this also works
293 // in modular arithmetic, doubling the correct bits in each step.
294 inverse *= 2 - denominator * inverse; // inverse mod 2^8
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
293 // in modular arithmetic, doubling the correct bits in each step.
294 inverse *= 2 - denominator * inverse; // inverse mod 2^8
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
293 // in modular arithmetic, doubling the correct bits in each step.
294 inverse *= 2 - denominator * inverse; // inverse mod 2^8
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
293 // in modular arithmetic, doubling the correct bits in each step.
294 inverse *= 2 - denominator * inverse; // inverse mod 2^8
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
294 inverse *= 2 - denominator * inverse; // inverse mod 2^8
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
294 inverse *= 2 - denominator * inverse; // inverse mod 2^8
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
294 inverse *= 2 - denominator * inverse; // inverse mod 2^8
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
299 inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
299 inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
295 inverse *= 2 - denominator * inverse; // inverse mod 2^16
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
299 inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
299 inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
299 inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
296 inverse *= 2 - denominator * inverse; // inverse mod 2^32
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
299 inverse *= 2 - denominator * inverse; // inverse mod 2^256
```

UNKNOWN Arithmetic operation "*=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
299 inverse *= 2 - denominator * inverse; // inverse mod 2^256
300
301 // Because the division is now exact we can divide by multiplying with the modular inverse of denominator.
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
299 inverse *= 2 - denominator * inverse; // inverse mod 2^256
300
301 // Because the division is now exact we can divide by multiplying with the modular inverse of denominator.
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
297 inverse *= 2 - denominator * inverse; // inverse mod 2^64
298 inverse *= 2 - denominator * inverse; // inverse mod 2^128
299 inverse *= 2 - denominator * inverse; // inverse mod 2^256
300
301 // Because the division is now exact we can divide by multiplying with the modular inverse of denominator.
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
303 // less than 2^256, this is the final result. We don't need to compute the high bits of the result and prod1
304 // is no longer required.
305 result = prod0 * inverse;
306 return result;
307 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
314 uint256 result = mulDiv(x, y, denominator);
315 if (rounding == Rounding.Up && mulmod(x, y, denominator) > 0) {
316 result += 1;
317 }
318 return result;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
346 // into the expected uint128 result.
347 unchecked {
348 result = (result + a / result) >> 1;
349 result = (result + a / result) >> 1;
350 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
346 // into the expected uint128 result.
347 unchecked {
348 result = (result + a / result) >> 1;
349 result = (result + a / result) >> 1;
350 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
347 unchecked {
348 result = (result + a / result) >> 1;
349 result = (result + a / result) >> 1;
350 result = (result + a / result) >> 1;
351 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
347 unchecked {
348 result = (result + a / result) >> 1;
349 result = (result + a / result) >> 1;
350 result = (result + a / result) >> 1;
351 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
348 result = (result + a / result) >> 1;
349 result = (result + a / result) >> 1;
350 result = (result + a / result) >> 1;
351 result = (result + a / result) >> 1;
352 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
348 result = (result + a / result) >> 1;
349 result = (result + a / result) >> 1;
350 result = (result + a / result) >> 1;
351 result = (result + a / result) >> 1;
352 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
349 result = (result + a / result) >> 1;
350 result = (result + a / result) >> 1;
351 result = (result + a / result) >> 1;
352 result = (result + a / result) >> 1;
353 result = (result + a / result) >> 1;
```

UNKNOWN

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
349 result = (result + a / result) >> 1;
350 result = (result + a / result) >> 1;
351 result = (result + a / result) >> 1;
352 result = (result + a / result) >> 1;
353 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
350 result = (result + a / result) >> 1;
351 result = (result + a / result) >> 1;
352 result = (result + a / result) >> 1;
353 result = (result + a / result) >> 1;
354 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
350 result = (result + a / result) >> 1;
351 result = (result + a / result) >> 1;
352 result = (result + a / result) >> 1;
353 result = (result + a / result) >> 1;
354 result = (result + a / result) >> 1;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
351 result = (result + a / result) >> 1;
352 result = (result + a / result) >> 1;
353 result = (result + a / result) >> 1;
354 result = (result + a / result) >> 1;
355 return min(result, a / result);
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
351 result = (result + a / result) >> 1;
352 result = (result + a / result) >> 1;
353 result = (result + a / result) >> 1;
354 result = (result + a / result) >> 1;
355 return min(result, a / result);
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
352 result = (result + a / result) >> 1;
353 result = (result + a / result) >> 1;
354 result = (result + a / result) >> 1;
355 return min(result, a / result);
356 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
352 result = (result + a / result) >> 1;
353 result = (result + a / result) >> 1;
354 result = (result + a / result) >> 1;
355 return min(result, a / result);
356 }
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
353 result = (result + a / result) >> 1;
354 result = (result + a / result) >> 1;
355 return min(result, a / result);
356 }
357 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
363 unchecked {
364 uint256 result = sqrt(a);
365 return result + (rounding == Rounding.Up && result * result < a ? 1 : 0);
366 }
367 }
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
363 unchecked {
364 uint256 result = sqrt(a);
365 return result + (rounding == Rounding.Up && result * result < a ? 1 : 0);
366 }
367 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
376 if (value >> 128 > 0) {
377 value >>= 128;
378 result += 128;
379 }
380 if (value >> 64 > 0) {
381 value >>= 64;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
380 if (value >> 64 > 0) {
381 value >>= 64;
382 result += 64;
383 }
384 if (value >> 32 > 0) {
385 value >>= 32;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
384 if (value >> 32 > 0) {
385 value >>= 32;
386 result += 32;
387 }
388 if (value >> 16 > 0) {
389 value >>= 16;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
388 if (value >> 16 > 0) {
389 value >>= 16;
390 result += 16;
391 }
392 if (value >> 8 > 0) {
393 value >>= 8;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
392 if (value >> 8 > 0) {
393 value >>= 8;
394 result += 8;
395 }
396 if (value >> 4 > 0) {
397 value >>= 4;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
396 if (value >> 4 > 0) {
397 value >>= 4;
398 result += 4;
399 }
400 if (value >> 2 > 0) {
401 value >>= 2;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
400 if (value >> 2 > 0) {
401 value >>= 2;
402 result += 2;
403 }
404 if (value >> 1 > 0) {
405 result += 1;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
403 }
404 if (value >> 1 > 0) {
405 result += 1;
406 }
407 }
408 return result;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
416 unchecked {
417 uint256 result = log2(value);
418 return result + (rounding == Rounding.Up && 1 << result < value ? 1 : 0);
419 }
420 }
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
427 uint256 result = 0;
428 unchecked {
429 if (value >= 10 ** 64) {
430 value /= 10 ** 64; value /= 10 ** 64;
431 result += 64;
432 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
428 unchecked {
429 if (value >= 10 ** 64) {
430 value /= 10 ** 64;
431 result += 64;
432 }
433 if (value >= 10 ** 32) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
428 unchecked {
429 if (value >= 10 ** 64) {
430 value /= 10 ** 64;
431 result += 64;
432 }
433 if (value >= 10 ** 32) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
429 if (value >= 10 ** 64) {
430 value /= 10 ** 64;
431 result += 64;
432 }
433 if (value >= 10 ** 32) {
434 value /= 10 ** 32;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
431 result += 64;
432 }
433 if (value >= 10 ** 32) {
434 value /= 10 ** 32; value /= 10 ** 32;
435 result += 32;
436 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
432 }
433 if (value >= 10 ** 32) {
434 value /= 10 ** 32;
435 result += 32;
436 }
437 if (value >= 10 ** 16) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
432 }
433 if (value >= 10 ** 32) {
434 value /= 10 ** 32;
435 result += 32;
436 }
437 if (value >= 10 ** 16) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
433 if (value >= 10 ** 32) {
434 value /= 10 ** 32;
435 result += 32;
436 }
437 if (value >= 10 ** 16) {
438 value /= 10 ** 16;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
435 result += 32;
436 }
437 if (value >= 10 ** 16) {
438 value /= 10 ** 16; value /= 10 ** 16;
439 result += 16;
440 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
436 }
437 if (value >= 10 ** 16) {
438 value /= 10 ** 16;
439 result += 16;
440 }
441 if (value >= 10 ** 8) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
436 }
437 if (value >= 10 ** 16) {
438 value /= 10 ** 16;
439 result += 16;
440 }
441 if (value >= 10 ** 8) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
437 if (value >= 10 ** 16) {
438 value /= 10 ** 16;
439 result += 16;
440 }
441 if (value >= 10 ** 8) {
442 value /= 10 ** 8;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
439 result += 16;
440 }
441 if (value >= 10 ** 8) {
442 value /= 10 ** 8; value /= 10 ** 8;
443 result += 8;
444 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
440 }
441 if (value >= 10 ** 8) {
442 value /= 10 ** 8;
443 result += 8;
444 }
445 if (value >= 10 ** 4) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
440 }
441 if (value >= 10 ** 8) {
442 value /= 10 ** 8;
443 result += 8;
444 }
445 if (value >= 10 ** 4) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
441 if (value >= 10 ** 8) {
442 value /= 10 ** 8;
443 result += 8;
444 }
445 if (value >= 10 ** 4) {
446 value /= 10 ** 4;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
443 result += 8;
444 }
445 if (value >= 10 ** 4) {
446 value /= 10 ** 4; value /= 10 ** 4;
447 result += 4;
448 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
444 }
445 if (value >= 10 ** 4) {
446 value /= 10 ** 4;
447 result += 4;
448 }
449 if (value >= 10 ** 2) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
444 }
445 if (value >= 10 ** 4) {
446 value /= 10 ** 4;
447 result += 4;
448 }
449 if (value >= 10 ** 2) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
445 if (value >= 10 ** 4) {
446 value /= 10 ** 4;
447 result += 4;
448 }
449 if (value >= 10 ** 2) {
450 value /= 10 ** 2;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
447 result += 4;
448 }
449 if (value >= 10 ** 2) {
450 value /= 10 ** 2; value /= 10 ** 2;
451 result += 2;
452 }
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
448 }
449 if (value >= 10 ** 2) {
450 value /= 10 ** 2;
451 result += 2;
452 }
453 if (value >= 10 ** 1) {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
448 }
449 if (value >= 10 ** 2) {
450 value /= 10 ** 2;
451 result += 2;
452 }
453 if (value >= 10 ** 1) {
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
449 if (value >= 10 ** 2) {
450 value /= 10 ** 2;
451 result += 2;
452 }
453 if (value >= 10 ** 1) {
454 result += 1;
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
451 result += 2;
452 }
453 if (value >= 10 ** 1) {
454 result += 1; result += 1;
455 }
456 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
452 }
453 if (value >= 10 ** 1) {
454 result += 1;
455 }
456 }
457 return result;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
465 unchecked {
466 uint256 result = log10(value);
467 return result + (rounding == Rounding.Up && 10 ** result < value ? 1 : 0);
468 }
469 }
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
465 unchecked {
466 uint256 result = log10(value);
467 return result + (rounding == Rounding.Up && 10 ** result < value ? 1 : 0);
468 }
469 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
480 if (value >> 128 > 0) {
481 value >>= 128;
482 result += 16;
483 }
484 if (value >> 64 > 0) {
485 value >>= 64;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
484 if (value >> 64 > 0) {
485 value >>= 64;
486 result += 8;
487 }
488 if (value >> 32 > 0) {
489 value >>= 32;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
488 if (value >> 32 > 0) {
489 value >>= 32;
490 result += 4;
491 }
492 if (value >> 16 > 0) {
493 value >>= 16;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
492 if (value >> 16 > 0) {
493 value >>= 16;
494 result += 2;
495 }
496 if (value >> 8 > 0) {
497 result += 1;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
495 }
496 if (value >> 8 > 0) {
497 result += 1;
498 }
499 }
500 return result;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
508 unchecked {
509 uint256 result = log256(value);
510 return result + (rounding == Rounding.Up && 1 << (result * 8) < value ? 1 : 0);
511 }
512 }
513 }
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
508 unchecked {
509 uint256 result = log256(value);
510 return result + (rounding == Rounding.Up && 1 << (result * 8) < value ? 1 : 0);
511 }
512 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
531 function toString(uint256 value) internal pure returns (string memory) {
532 unchecked {
533 uint256 length = Math.log10(value) + 1;
534 string memory buffer = new string(length);
535 uint256 ptr;
536 /// @solidity memory-safe-assembly
```

UNKNOWN Arithmetic operation "--" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
539 }
540 while (true) {
541 ptr--;
542 /// @solidity memory-safe-assembly
543 assembly {
544 mstore8(ptr, byte(mod(value, 10), _SYMBOLS))
```

UNKNOWN Arithmetic operation "/=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
544 mstore8(ptr, byte(mod(value, 10), _SYMBOLS))
545 }
546 value /= 10;
547 if (value == 0) break;
548 }
549 return buffer;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
556 function toHexString(uint256 value) internal pure returns (string memory) {
557 unchecked {
558 return toHexString(value, Math.log256(value) + 1);
559 }
560 }
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
564 */
565 function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
566 bytes memory buffer = new bytes(2 * length + 2);
567 buffer[0] = "0";
568 buffer[1] = "x";
569 for (uint256 i = 2 * length + 1; i > 1; --i) {
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
564 */
565 function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
566 bytes memory buffer = new bytes(2 * length + 2);
567 buffer[0] = "0";
568 buffer[1] = "x";
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
567 buffer[0] = "0";
568 buffer[1] = "x";
569 for (uint256 i = 2 * length + 1; i > 1; --i) {
570 buffer[i] = _SYMBOLS[value & 0xf];
571 value >>= 4;
```

UNKNOWN

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
567 buffer[0] = "0";
568 buffer[1] = "x";
569 for (uint256 i = 2 * length + 1; i > 1; --i) {
570 buffer[i] = _SYMBOLS[value & 0xf];
571 value >>= 4;
```

UNKNOWN

Arithmetic operation "--" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
567 buffer[0] = "0";
568 buffer[1] = "x";
569 for (uint256 i = 2 * length + 1; i > 1; --i) {
570 buffer[i] = _SYMBOLS[value & 0xf]; buffer[i] = _SYMBOLS[value & 0xf];
571 value >>= 4;
572 }
```

UNKNOWN

Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1021 // cannot realistically overflow on human timescales
1022 unchecked {
1023 ++i;
1024 }
1025 }
1026 }
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1399 uint quoteAmount = _multiply(quantity, BASE_TOKEN_DECIMALS, price, QUOTE_TOKEN_DECIMALS);
1400 _transferFrom(quoteToken, msg.sender, address(this), quoteAmount); // transfer quote token to this contract
1401 bids[msg.sender][price] += quantity;
1402 } else {
1403 _transferFrom(baseToken, msg.sender, address(this), quantity); // transfer base token to this contract
1404 asks[msg.sender][price] += quantity;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1402 } else {
1403 _transferFrom(baseToken, msg.sender, address(this), quantity); // transfer base token to this contract
1404 asks[msg.sender][price] += quantity;
1405 }
1406
1407 emit OrderCreated(msg.sender, side, price, quantity);
```

UNKNOWN Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1419
1420 uint remainningQuantity = quantity;
1421 for (uint i; i < makers.length; ++i) {
1422 require(makers[i] != address(0), "Invalid maker"); require(makers[i] != address(0), "Invalid maker");
1423 if (side == 0) {
1424 uint makerQuantity = asks[makers[i]][price];
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1430 } else {
1431 transferQuantity = makerQuantity;
1432 remainningQuantity = remainningQuantity - makerQuantity;
1433 }
1434 uint makerTransferQuantity = _multiply(
1435 transferQuantity,
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1439 );
1440 _transferFrom(quoteToken, msg.sender, makers[i], makerTransferQuantity); // transfer qoute token from taker to maker
1441 uint fee = (transferQuantity * tradeFee) / 10000;
1442 totalBaseFee += fee;
1443 uint remainningToTaker = transferQuantity - fee;
1444 _transfer(baseToken, msg.sender, remainningToTaker); // transfer base token from contract to maker
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1439 );
1440 _transferFrom(quoteToken, msg.sender, makers[i], makerTransferQuantity); // transfer qoute token from taker to maker
1441 uint fee = (transferQuantity * tradeFee) / 10000;
1442 totalBaseFee += fee;
1443 uint remainningToTaker = transferQuantity - fee;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1440 _transferFrom(quoteToken, msg.sender, makers[i], makerTransferQuantity); // transfer qoute token from taker to maker
1441 uint fee = (transferQuantity * tradeFee) / 10000;
1442 totalBaseFee += fee;
1443 uint remainningToTaker = transferQuantity - fee;
1444 _transfer(baseToken, msg.sender, remainningToTaker); // transfer base token from contract to maker
1445 asks[makers[i]][price] -= transferQuantity;
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1441 uint fee = (transferQuantity * tradeFee) / 10000;
1442 totalBaseFee += fee;
1443 uint remainningToTaker = transferQuantity - fee;
1444 _transfer(baseToken, msg.sender, remainningToTaker); // transfer base token from contract to maker
1445 asks[makers[i]][price] -= transferQuantity;
1446 if (asks[makers[i]][price] == 0) {
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1443 uint remainningToTaker = transferQuantity - fee;
1444 _transfer(baseToken, msg.sender, remainningToTaker); // transfer base token from contract to maker
1445 asks[makers[i]][price] -= transferQuantity;
1446 if (asks[makers[i]][price] == 0) {
1447 delete asks[makers[i]][price];
1448 emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1458 } else {
1459 transferQuantity = makerQuantity;
1460 remainningQuantity = remainningQuantity - makerQuantity;
1461 }
1462 _transferFrom(baseToken, msg.sender, makers[i], transferQuantity); // transfer base from taker to maker
1463 uint quantityWithoutFee = _multiply(transferQuantity, BASE_TOKEN_DECIMALS, price, QUOTE_TOKEN_DECIMALS);
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1462 _transferFrom(baseToken, msg.sender, makers[i], transferQuantity); // transfer base from taker to maker
1463 uint quantityWithoutFee = _multiply(transferQuantity, BASE_TOKEN_DECIMALS, price, QUOTE_TOKEN_DECIMALS);
1464 uint fee = (quantityWithoutFee * tradeFee) / 10000;
1465 totalQuoteFee += fee;
1466 uint remainningToTaker = quantityWithoutFee - fee;
1467 _transfer(quoteToken, msg.sender, remainningToTaker); // transfer usd from contract to taker
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1462 _transferFrom(baseToken, msg.sender, makers[i], transferQuantity); // transfer base from taker to maker
1463 uint quantityWithoutFee = _multiply(transferQuantity, BASE_TOKEN_DECIMALS, price, QUOTE_TOKEN_DECIMALS);
1464 uint fee = (quantityWithoutFee * tradeFee) / 10000;
1465 totalQuoteFee += fee;
1466 uint remainningToTaker = quantityWithoutFee - fee;
```

UNKNOWN Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1463 uint quantityWithoutFee = _multiply(transferQuantity, BASE_TOKEN_DECIMALS, price, QUOTE_TOKEN_DECIMALS);
1464 uint fee = (quantityWithoutFee * tradeFee) / 10000;
1465 totalQuoteFee += fee;
1466 uint remainningToTaker = quantityWithoutFee - fee;
1467 _transfer(quoteToken, msg.sender, remainningToTaker); // transfer usd from contract to taker
1468 bids[makers[i]][price] -= transferQuantity;
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1464 uint fee = (quantityWithoutFee * tradeFee) / 10000;
1465 totalQuoteFee += fee;
1466 uint remainningToTaker = quantityWithoutFee - fee;
1467 _transfer(quoteToken, msg.sender, remainningToTaker); // transfer usd from contract to taker
1468 bids[makers[i]][price] -= transferQuantity;
1469 if (bids[makers[i]][price] == 0) {
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1466 uint remainningToTaker = quantityWithoutFee - fee;
1467 _transfer(quoteToken, msg.sender, remainningToTaker); // transfer usd from contract to taker
1468 bids[makers[i]][price] -= transferQuantity;
1469 if (bids[makers[i]][price] == 0) {
1470 delete bids[makers[i]][price];
1471 emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1493 if (oldQuantity > desiredQuantity) {
1494 uint _quantity = _multiply(
1495 oldQuantity - desiredQuantity,
1496 BASE_TOKEN_DECIMALS,
1497 price,
1498 QUOTE_TOKEN_DECIMALS
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1501 } else if (oldQuantity < desiredQuantity) {
1502 uint _quantity = _multiply(
1503 desiredQuantity - oldQuantity,
1504 BASE_TOKEN_DECIMALS,
1505 price,
1506 QUOTE_TOKEN_DECIMALS
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1513 require(oldQuantity > 0, "Order does not exist");
1514 if (oldQuantity > desiredQuantity) {
1515 _transfer(baseToken, msg.sender, oldQuantity - desiredQuantity);
1516 } else if (oldQuantity < desiredQuantity) {
1517 _transferFrom(baseToken, msg.sender, address(this), desiredQuantity - oldQuantity); // transfer base token to this contract
1518 }
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1515 _transfer(baseToken, msg.sender, oldQuantity - desiredQuantity);
1516 } else if (oldQuantity < desiredQuantity) {
1517 _transferFrom(baseToken, msg.sender, address(this), desiredQuantity - oldQuantity); // transfer base token to this contract
1518 }
1519 asks[msg.sender][price] = desiredQuantity;
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1532 */
1533 function _multiply(uint x, uint8 xDecimals, uint y, uint8 yDecimals) private pure returns (uint) {
1534 uint prod = x * y;
1535 uint8 prodDecimals = xDecimals + yDecimals;
1536 if (prodDecimals < yDecimals) {
1537 return prod * (10 ** (yDecimals - prodDecimals));
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1533 function _multiply(uint x, uint8 xDecimals, uint y, uint8 yDecimals) private pure returns (uint) {
1534 uint prod = x * y;
1535 uint8 prodDecimals = xDecimals + yDecimals;
1536 if (prodDecimals < yDecimals) {
1537 return prod * (10 ** (yDecimals - prodDecimals));
1538 } else if (prodDecimals > yDecimals) {
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1535 uint8 prodDecimals = xDecimals + yDecimals;
1536 if (prodDecimals < yDecimals) {
1537 return prod * (10 ** (yDecimals - prodDecimals));
1538 } else if (prodDecimals > yDecimals) {
1539 return prod / (10 ** (prodDecimals - yDecimals));
1540 } else {
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1535 uint8 prodDecimals = xDecimals + yDecimals;
1536 if (prodDecimals < yDecimals) {
1537 return prod * (10 ** (yDecimals - prodDecimals));
1538 } else if (prodDecimals > yDecimals) {
1539 return prod / (10 ** (prodDecimals - yDecimals));
1540 } else {
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1535 uint8 prodDecimals = xDecimals + yDecimals;
1536 if (prodDecimals < yDecimals) {
1537 return prod * (10 ** (yDecimals - prodDecimals));
1538 } else if (prodDecimals > yDecimals) { } else if (prodDecimals > yDecimals) {
1539 return prod / (10 ** (prodDecimals - yDecimals));
1540 } else {
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1537 return prod * (10 ** (yDecimals - prodDecimals));
1538 } else if (prodDecimals > yDecimals) {
1539 return prod / (10 ** (prodDecimals - yDecimals));
1540 } else {
1541 return prod;
1542 }
```

UNKNOWN Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1537 return prod * (10 ** (yDecimals - prodDecimals));
1538 } else if (prodDecimals > yDecimals) {
1539 return prod / (10 ** (prodDecimals - yDecimals));
1540 } else {
1541 return prod;
1542 }
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1537 return prod * (10 ** (yDecimals - prodDecimals));
1538 } else if (prodDecimals > yDecimals) {
1539 return prod / (10 ** (prodDecimals - yDecimals));
1540 } else { } else {
1541 return prod;
1542 }
```

UNKNOWN

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1569 require(amount > 0, "Zero amount");
1570 if (msg.value > 0) {
1571 _counter += amount;
1572 require(_counter <= msg.value, "Incorrect amount");
1573 if (token == weth) {
1574 IWETH(weth).deposit{value: amount}();
```

UNKNOWN

Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/DexillaExchangeV4.sol

Locations

```
222 function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
223 // (a + b - 1) / b can overflow on addition, so we distribute.
224 return a == 0 ? 0 : (a - 1) / b + 1;
225 }
```

MEDIUM

Potential incorrect constructor name "selfPermit".

Until Solidity version 0.4.21 the constructor can only be defined as a function with the exact same name as the contract class. The function "selfPermit" looks similar to a constructor for "SelfPermit" but is not a constructor. Please rename to avoid confusion.

SWC-118

Source file

contracts/DexillaExchangeV4.sol

Locations

```
914
915 abstract contract SelfPermit {
916 function selfPermit(address token, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s) public payable {
917 IERC20Permit(token).permit(msg.sender, address(this), value, deadline, v, r, s);
918 }
919
920 function selfPermit2(address token, uint value, uint deadline, bytes calldata signature) public payable {
921 IERC20Permit2(token).permit2(msg.sender, address(this), value, deadline, signature);
922 }
```

MEDIUM

Potential incorrect constructor name "multicall".

SWC-118

Until Solidity version 0.4.21 the constructor can only be defined as a function with the exact same name as the contract class. The function "multicall" looks similar to a constructor for "Multicall" but is not a constructor. Please rename to avoid confusion.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1002 /// License-Identifier: GPL-2.0-or-later
1003 abstract contract Multicall {
1004 function multicall(bytes[] calldata data) public payable returns (bytes[] memory results) {
1005 results = new bytes[](data.length);
1006
1007 for (uint i; i < data.length; ) {
1008 (bool success, bytes memory result) = address(this).delegatecall(data[i]);
1009
1010 if (!success) {
1011 // Next 5 lines from https://ethereum.stackexchange.com/a/83577
1012 if (result.length < 68) revert();
1013 assembly {
1014 result := add(result, 0x04)
1015 }
1016 revert(abi.decode(result, (string)));
1017 }
1018
1019 results[i] = result;
1020
1021 // cannot realistically overflow on human timescales
1022 unchecked {
1023 ++i;
1024 }
1025 }
1026 }
1027 }
1028
1029 // File contracts/interfaces/IWETH.sol// File contracts/interfaces/IWETH.sol
1030
1031 pragma solidity ^0.8.0;
```

LOW

A floating pragma is set.

SWC-103

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
7 // OpenZeppelin Contracts v4.4.1 (access/IAccessControl.sol)
8
9 pragma solidity ^0.8.0;
10
11 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
97  // OpenZeppelin Contracts v4.4.1 (utils/Context.sol)
98
99  pragma solidity ^0.8.0;
100
101 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
123 // OpenZeppelin Contracts v4.4.1 (utils/introspection/IERC165.sol)
124
125 pragma solidity ^0.8.0;
126
127 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
150 // OpenZeppelin Contracts v4.4.1 (utils/introspection/ERC165.sol)
151
152 pragma solidity ^0.8.0;
153
154 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
179 // OpenZeppelin Contracts (last updated v4.8.0) (utils/math/Math.sol)
180
181 pragma solidity ^0.8.0;
182
183 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
517 // OpenZeppelin Contracts (last updated v4.8.0) (utils/Strings.sol)
518
519 pragma solidity ^0.8.0;
520
521 /**
522 * @dev String operations.
523 */
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
587 // OpenZeppelin Contracts (last updated v4.8.0) (access/AccessControl.sol)
588
589 pragma solidity ^0.8.0;
590
591 /**
592 * @dev Contract module that allows children to implement role-based access
593 * control mechanisms. This is a lightweight version that doesn't allow enumerating role
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.5.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
829 // File contracts/interfaces/token/IERC20Base.sol
830
831 pragma solidity ^0.5.0;
832
833 interface IERC20Base {
834 function totalSupply() external view returns (uint);
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.5.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
850 // File contracts/interfaces/token/IERC20.sol
851
852 pragma solidity ^0.5.0;
853
854 interface IERC20 is IERC20Base {
855 function name() external view returns (string memory);
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.5.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
862 // File contracts/interfaces/token/IERC20Permit.sol
863
864 pragma solidity ^0.5.0;
865
866 interface IERC20Permit is IERC20 {
867 function permit(address owner, address spender, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s) external;
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
874 // File contracts/interfaces/token/IERC20Permit2.sol
875
876 pragma solidity >=0.5.0;
877
878 interface IERC20Permit2 is IERC20Permit {
879 function permit2(address owner, address spender, uint amount, uint deadline, bytes calldata signature) external;
880 }
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
882 // File contracts/interfaces/token/IERC20PermitAllowed.sol
883
884 pragma solidity >=0.5.0;
885
886 /// @title Interface for permit
887 /// @notice Interface used by DAI/CHAI for permit
888 interface IERC20PermitAllowed {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
911 // File contracts/abstract/SelfPermit.sol
912
913 pragma solidity ^0.8.0;
914
915 abstract contract SelfPermit {
916 function selfPermit(address token, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s) public payable {
917 IERC20Permit(token).permit(msg.sender, address(this), value, deadline, v, r, s);
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
927 // OpenZeppelin Contracts (last updated v4.8.0) (security/ReentrancyGuard.sol)
928
929 pragma solidity ^0.8.0;
930
931 /**
932 * @dev Contract module that helps prevent reentrant calls to a function.
933 *
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
996 // File contracts/abstract/Multicall.sol
997
998 pragma solidity ^0.8.0;
999
1000 /// @notice Helper utility that enables calling multiple local methods in a single call.
1001 /// @author Modified from Uniswap (https://github.com/Uniswap/v3-periphery/blob/main/contracts/base/Multicall.sol)
1002 /// License-Identifier: GPL-2.0-or-later
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1029 // File contracts/interfaces/IWETH.sol
1030
1031 pragma solidity ^0.8.0;
1032
1033 interface IWETH {
1034 function deposit() external payable;
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1043 // File contracts/libraries/TransferHelper.sol
1044
1045 pragma solidity ^0.8.0;
1046
1047 /// @dev The ETH transfer has failed.
1048 error ETHTransferFailed();
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
565 function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
566 bytes memory buffer = new bytes(2 * length + 2);
567 buffer[0] = "0";
568 buffer[1] = "x";
569 for (uint256 i = 2 * length + 1; i > 1; --i) {
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

contracts/DexillaExchangeV4.sol

Locations

```
566 bytes memory buffer = new bytes(2 * length + 2);
567 buffer[0] = "0";
568 buffer[1] = "x";
569 for (uint256 i = 2 * length + 1; i > 1; --i) {
570 buffer[i] = _SYMBOLS[value & 0xf];
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
568 buffer[1] = "x";
569 for (uint256 i = 2 * length + 1; i > 1; --i) {
570 buffer[i] = _SYMBOLS[value & 0xf];
571 value >>= 4;
572 }
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
568 buffer[1] = "x";
569 for (uint256 i = 2 * length + 1; i > 1; --i) {
570 buffer[i] = _SYMBOLS[value & 0xf];
571 value >>= 4;
572 }
573 require(value == 0, "Strings: hex length insufficient");
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1006
1007 for (uint i; i < data.length; ) {
1008 (bool success, bytes memory result) = address(this).delegatecall(data[i]);
1009
1010 if (!success) { if (!success) {
1011 // Next 5 lines from https://ethereum.stackexchange.com/a/83577
1012 if (result.length < 68) revert();
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1017 }
1018
1019 results[i] = result;
1020
1021 // cannot realistically overflow on human timescales
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1420 uint remainningQuantity = quantity;
1421 for (uint i; i < makers.length; ++i) {
1422 require(makers[i] != address(0), "Invalid maker");
1423 if (side == 0) {
1424 uint makerQuantity = asks[makers[i]][price];
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1422 require(makers[i] != address(0), "Invalid maker");
1423 if (side == 0) {
1424 uint makerQuantity = asks[makers[i]][price];
1425 if (makerQuantity == 0) continue;
1426 uint transferQuantity;
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1438 QUOTE_TOKEN_DECIMALS
1439 );
1440 _transferFrom(quoteToken, msg.sender, makers[i], makerTransferQuantity); // transfer qoute token from taker to maker
1441 uint fee = (transferQuantity * tradeFee) / 10000;
1442 totalBaseFee += fee;
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1443 uint remainningToTaker = transferQuantity - fee;
1444 _transfer(baseToken, msg.sender, remainningToTaker); // transfer base token from contract to maker
1445 asks[makers[i]][price] -= transferQuantity;
1446 if (asks[makers[i]][price] == 0) {
1447 delete asks[makers[i]][price];
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1444 _transfer(baseToken, msg.sender, remainningToTaker); // transfer base token from contract to maker
1445 asks[makers[i]][price] -= transferQuantity;
1446 if (asks[makers[i]][price] == 0) {
1447 delete asks[makers[i]][price];
1448 emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1445 asks[makers[i]][price] -= transferQuantity;
1446 if (asks[makers[i]][price] == 0) {
1447 delete asks[makers[i]][price];
1448 emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
1449 }
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1446 if (asks[makers[i]][price] == 0) {
1447 delete asks[makers[i]][price];
1448 emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
1449 }
1450 emit OrderExecuted(makers[i], msg.sender, side, price, transferQuantity, fee);
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1448 emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
1449 }
1450 emit OrderExecuted(makers[i], msg.sender, side, price, transferQuantity, fee);
1451 } else {
1452 uint makerQuantity = bids[makers[i]][price];
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1450 emit OrderExecuted(makers[i], msg.sender, side, price, transferQuantity, fee);
1451 } else {
1452 uint makerQuantity = bids[makers[i]][price];
1453 if (makerQuantity == 0) continue;
1454 uint transferQuantity;
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1460 remainningQuantity = remainningQuantity - makerQuantity;
1461 }
1462 _transferFrom(baseToken, msg.sender, makers[i], transferQuantity); // transfer base from taker to maker
1463 uint quantityWithoutFee = _multiply(transferQuantity, BASE_TOKEN_DECIMALS, price, QUOTE_TOKEN_DECIMALS);
1464 uint fee = (quantityWithoutFee * tradeFee) / 10000;
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1466 uint remainningToTaker = quantityWithoutFee - fee;
1467 _transfer(quoteToken, msg.sender, remainningToTaker); // transfer usd from contract to taker
1468 bids[makers[i]][price] -= transferQuantity;
1469 if (bids[makers[i]][price] == 0) {
1470 delete bids[makers[i]][price];
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1467 _transfer(quoteToken, msg.sender, remainningToTaker); // transfer usd from contract to taker
1468 bids[makers[i]][price] -= transferQuantity;
1469 if (bids[makers[i]][price] == 0) {
1470 delete bids[makers[i]][price];
1471 emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1468 bids[makers[i]][price] -= transferQuantity;
1469 if (bids[makers[i]][price] == 0) {
1470 delete bids[makers[i]][price];
1471 emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
1472 }
```

UNKNOWN Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1469 if (bids[makers[i]][price] == 0) {
1470 delete bids[makers[i]][price];
1471 emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
1472 }
1473 emit OrderExecuted(makers[i], msg.sender, side, price, transferQuantity, fee);
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

contracts/DexillaExchangeV4.sol

Locations

```
1471 | emit OrderCanceled(makers[i], side ^ 1, price, transferQuantity);
1472 | }
1473 | emit OrderExecuted(makers[i], msg.sender, side, price, transferQuantity, fee);
1474 | }
1475 | if (remainningQuantity == 0) break;
```