**Xz vs ClamAV**

**Notes**

* ClamaAV uses both XZ and 7z which both use the same compression/decompression algorithm LZMA, while XZ Utils uses LZMA directly
* XZ seems to use a lot more bitwise operations (&, ||, >>, <<)
* Many error checks skipped in ClamAV due to defining: ifdef SYMBEX\_DISABLE\_CRCCHECK (vli\_decoder.c vs nothing), making it skip checks on key byte positions from the input that would have triggered an error.
* While this seems to be the root of most errors, it is also important to note that there is also this ifdef SYMBEX\_DISABLE\_CRCCHECK in some of XZ Util’s deeper libraries. !!!!!!!!!! VERY INTERESTING !!!!!!!!!, XZ also has its CRCCHECKs disabled, but there are less of them.
* However, even with SYMBEX being not defined, it seems like there are still less checks for ClamAV then XZ Utils, which could be due to the fact that ClamAV is an antivirus scanner and is thus much more liberal in what it is willing to decompress, which would allow it to better detect potential viruses that could be embedded inside the compressed file.
* Potentially possible that ClamAV tries to use a mix of XZ Utils and 7z in order to not be held accountable by opensource software laws/rules, which could have been another source of the discrepancies found.
* The inputs are a combination of 52 bytes, and the positions I refer to in this document are the positions of the bytes that are triggering an error.
* Lastly, many of the inputs trigger discrepancies in the same code area in case you want to try debugging the inputs that PATHDIFF found.
* !!! ClamAV accepts only inputs of type .xz while XZ Utils accepts inputs of either .xz or lzma types.
* NEW FINDING: The bitwise operations in XZ are actually also CRC checks, potentially causing some “false” discrepancies as they were not correctly disabled in XZ while they were disabled in ClamAV. This seems to be a recurring problem in Differential Testing Tools as sometimes, parts of codes must be omitted (disabled) in order for the SAT solver to work, but it’s very difficult to correctly disable codes such to ensure that only the desired parts are omitted while not creating “artificial” or “false” discrepancies.

**Equivalencies XZ vs ClamAV**

**1.** **main.c** : coder\_run/ run(args.arg\_names[i])

==

**clamav-0.99.2-xz.c** : CUSTOM\_xz\_parse(data, size = 52)

**2.** **coder.c** : coder\_init/ ret = lzma\_code(&strm, LZMA\_RUN)

==

**scanners.c :** rc = cli\_XzDecode(&ctx->strm)

**3.** **common.c** : extern LZMA\_API(lzma\_ret) lzma\_code(lzma\_stream \*strm, lzma\_action action)

==

**xz\_iface.c** : int cli\_XzDecode(struct CLI\_XZ \*XZ)

**4.** **stream\_decoder.c** : static lzma\_ret stream\_decode(lzma\_coder \*coder, const lzma\_allocator \*allocator, const uint8\_t \*restrict in, size\_t \*restrict in\_pos, size\_t in\_size, uint8\_t \*restrict out, size\_t \*restrict out\_pos, size\_t out\_size, lzma\_action action)

==

**XzDec.c** : SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)

**Command Shortcuts to Run Programs**

1. gdb --args build/xz/clamav-0.99.2/clamav-0.99.2-xz data/xz/20211125\_135555\_xz-5.2.2-clamav-0.99.2\_tgt-1\_all/xz-5.2.2-clamav-0.99.2-tgt-1-ivtpos-49-subfuzz-1-dis-1.xz

2. gdb --args build/xz/xz-5.2.2/xz-5.2.2 -dkc data/xz/20211125\_135555\_xz-5.2.2-clamav-0.99.2\_tgt-1\_all/xz-5.2.2-clamav-0.99.2-tgt-1-ivtpos-49-subfuzz-1-dis-1.xz

3. gdb --args build/xz/clamav-0.99.2/clamav-0.99.2-xz data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-100-subfuzz-1-dis-1.xz

4. gdb --args build/xz/xz-5.2.2/xz-5.2.2 -dkc data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-104-subfuzz-1-dis-1.xz

5. gdb --args build/xz/clamav-0.99.2/clamav-0.99.2-xz data/xz/20220510\_115409\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-0\_all/testcases/id-0000000701-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-0-ivtpos-7-subfuzz-1-dis-1.xz

6. gdb --args build/xz/clamav-0.99.2/clamav-0.99.2-xz data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000001-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-0-subfuzz-1-dis-3.xz

7. gdb --args build/xz/clamav-0.99.2/clamav-0.99.2-xz data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000101-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-1-subfuzz-1-dis-3.xz

**General Format:**

* gdb --args build/xz/clamav-0.99.2/clamav-0.99.2-xz (…) or gdb --args build/xz/xz-5.2.2/xz-5.2.2 -dkc (…)
* (…) => @@ (path)

**Expanded Case Legend**

* : Equivalent Code
* **\*\*** : Due to disabling CRC Checks

|  |  |  |
| --- | --- | --- |
| @@ (path) | XZ Utils | ClamAV |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-100-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  **For ClamAV:**  // Skipped Check at lines 798, 827, 837 in case: XZ\_STATE\_STREAM\_INDEX of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  à  #ifdef SYMBEX\_DISABLE\_CRCCHECK            src++;            p->pos++;  #else            if (\*src++ != p->buf[p->pos++])              return SZ\_ERROR\_CRC;  #endif  vs  **For XZ:**  // Sanity check on accurate block count at line 207 in index\_hash.c and function:  extern LZMA\_API(lzma\_ret)  lzma\_index\_hash\_decode(lzma\_index\_hash \*index\_hash, const uint8\_t \*in,          size\_t \*in\_pos, size\_t in\_size)  !!!  // The count must match the count of the Blocks decoded.          if (index\_hash->remaining != index\_hash->blocks.count)              return LZMA\_DATA\_ERROR;  // Root of discrepancy is still due to the crcchecks (similar to 98 and 99), where clamAV skips scanning certain bytes by default as opposed to XZ’s vli\_decoder.c  // At index\_hash.c line 201, which uses vli\_decoder.c  ret = lzma\_vli\_decode(&index\_hash->remaining,                  &index\_hash->pos, in, in\_pos, in\_size);          if (ret != LZMA\_STREAM\_END)              goto out;          // The count must match the count of the Blocks decoded.          if (index\_hash->remaining != index\_hash->blocks.count)              return LZMA\_DATA\_ERROR;  // here the variable &index\_hash->remaining refers to the adress of index\_hash->remaining, which is changed in the lzma\_vli\_decode function of vli\_decoder.c at line 49:  // Read the next byte. Use a temporary variable so that we          // can update \*in\_pos immediately.          const uint8\_t byte = in[\*in\_pos];          ++\*in\_pos;          // Add the newly read byte to \*vli.          \*vli += (lzma\_vli)(byte & 0x7F) << (\*vli\_pos \* 7);  // \*Note\*: Vli == &index\_hash->remaining  // Because of disabling crc checks, ClamAV skips over checking these bytes. | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-104-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // \*in\_pos == 34  // This Discrepancy is not due to CRC, but because of an actual difference in implementation  See Additional Info Section and ivtpos 106 for more information | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-105-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // \*in\_pos == 34  // This Discrepancy is not due to CRC, but because of an actual difference in implementation  See Additional Info Section and ivtpos 106 for more information | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-106-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  **For ClamAV:**  // Missing Check at lines 808-819 in case: XZ\_STATE\_STREAM\_INDEX of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  if (p->indexPos < p->indexSize)            {              UInt64 cur = p->indexSize - p->indexPos;              if (srcRem > cur)                srcRem = (SizeT)cur;              p->crc = CrcUpdate(p->crc, src, srcRem);              if ((p->sha))                  cl\_update\_hash(p->sha, (void \*)src, srcRem);              (\*srcLen) += srcRem;              src += srcRem;              p->indexPos += srcRem;            }  vs  **For XZ:**  // Sanity check on accurate block count at line 207 in index\_hash.c and function:  extern LZMA\_API(lzma\_ret)  lzma\_index\_hash\_decode(lzma\_index\_hash \*index\_hash, const uint8\_t \*in,          size\_t \*in\_pos, size\_t in\_size)  !!!  // The count must match the count of the Blocks decoded.          if (index\_hash->remaining != index\_hash->blocks.count)              return LZMA\_DATA\_ERROR;  // At index\_hash.c line 201, which uses vli\_decoder.c  ret = lzma\_vli\_decode(&index\_hash->remaining,                  &index\_hash->pos, in, in\_pos, in\_size);          if (ret != LZMA\_STREAM\_END)              goto out;          // The count must match the count of the Blocks decoded.          if (index\_hash->remaining != index\_hash->blocks.count)              return LZMA\_DATA\_ERROR;  // here the variable &index\_hash->remaining refers to the adress of index\_hash->remaining, which is changed in the lzma\_vli\_decode function of vli\_decoder.c at line 49:  // Read the next byte. Use a temporary variable so that we          // can update \*in\_pos immediately.          const uint8\_t byte = in[\*in\_pos];          ++\*in\_pos;          // Add the newly read byte to \*vli.          \*vli += (lzma\_vli)(byte & 0x7F) << (\*vli\_pos \* 7);  // \*Note\*: Vli == &index\_hash->remaining  // See Additional Info Section | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-110-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // \*in\_pos == 35  **For ClamAV:**  // Completely skipped checking over byte 35  // See Additional Info Section and ivtpos 104/105/10  **For XZ:** | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-111-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // same as 110 | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-114-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| **For ClamAV:**  // Skipped Check at line 839 in case: XZ\_STATE\_STREAM\_INDEX of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  #ifdef SYMBEX\_DISABLE\_CRCCHECK  #else              if ((p->sha))                  cl\_finish\_hash(p->sha, digest);              if (memcmp(digest, p->shaDigest, SHA256\_DIGEST\_SIZE) != 0)                return SZ\_ERROR\_CRC;  #endif  vs  **For XZ:**  // Sanity check on block sizes at line 252 in index\_hash.c and function:  extern LZMA\_API(lzma\_ret)  lzma\_index\_hash\_decode(lzma\_index\_hash \*index\_hash, const uint8\_t \*in, size\_t \*in\_pos, size\_t in\_size)  // Verify that we don't go over the known sizes.  // Note that this validation is simpler than the one used in lzma\_index\_hash\_append(), because here we know that values in index\_hash->blocks are already validated and we are fine as long as we don't exceed them in index\_hash->records.  !!!              if (index\_hash->blocks.blocks\_size                      < index\_hash->records.blocks\_size                      || index\_hash->blocks.uncompressed\_size                      < index\_hash->records.uncompressed\_size                      || index\_hash->blocks.index\_list\_size                      < index\_hash->records.index\_list\_size)                  return LZMA\_DATA\_ERROR;  // At index\_hash.c line 219, which uses vli\_decoder.c  case SEQ\_UNPADDED:  case SEQ\_UNCOMPRESSED: {          lzma\_vli \*size = index\_hash->sequence == SEQ\_UNPADDED                  ? &index\_hash->unpadded\_size                  : &index\_hash->uncompressed\_size;          ret = lzma\_vli\_decode(size, &index\_hash->pos,                  in, in\_pos, in\_size);  // here the variable size refers to the adress of unpadded\_size, which is changed in the lzma\_vli\_decode function of vli\_decoder.c at line 49:  // Read the next byte. Use a temporary variable so that we          // can update \*in\_pos immediately.          const uint8\_t byte = in[\*in\_pos];          ++\*in\_pos;          // Add the newly read byte to \*vli.          \*vli += (lzma\_vli)(byte & 0x7F) << (\*vli\_pos \* 7);  // \*Note\*: Vli == size  // ClamAV skips over checking these bytes due to the presence of a SYMBEX DISABLE CRCCHECK | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-118-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // Nearly same as 114  For ClamAV:  // Skipped Check at lines 798, 827, 837 in case: XZ\_STATE\_STREAM\_INDEX of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  à  #ifdef SYMBEX\_DISABLE\_CRCCHECK            src++;            p->pos++;  #else            if (\*src++ != p->buf[p->pos++])              return SZ\_ERROR\_CRC;  #endif  AND  #ifdef SYMBEX\_DISABLE\_CRCCHECK  #else              if (b != 0)                return SZ\_ERROR\_CRC;  #endif  AND  #ifdef SYMBEX\_DISABLE\_CRCCHECK  #else              if ((p->sha))                  cl\_finish\_hash(p->sha, digest);              if (memcmp(digest, p->shaDigest, SHA256\_DIGEST\_SIZE) != 0)                return SZ\_ERROR\_CRC;  vs  For XZ:  // Sanity check on block sizes at line 286 instead of 252 in index\_hash.c and function:  extern LZMA\_API(lzma\_ret)  lzma\_index\_hash\_decode(lzma\_index\_hash \*index\_hash, const uint8\_t \*in, size\_t \*in\_pos, size\_t in\_size)  // Just constantly rechecking that the block sizes are consistent with the records of their sizes.  // Compare the sizes.          if (index\_hash->blocks.blocks\_size                  != index\_hash->records.blocks\_size                  || index\_hash->blocks.uncompressed\_size                  != index\_hash->records.uncompressed\_size                  || index\_hash->blocks.index\_list\_size                  != index\_hash->records.index\_list\_size)              return LZMA\_DATA\_ERROR; | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-125-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  **For ClamAV:**  // Skipped Check at line 888 in case: XZ\_STATE\_STREAM\_FOOTER of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  à  #ifdef SYMBEX\_DISABLE\_CRCCHECK  #else            if (!Xz\_CheckFooter(p->streamFlags, p->indexSize, p->buf))              return SZ\_ERROR\_CRC;  #endif  vs  **For XZ:**  // Checking to see if the inputs matches the record at line 65 in stream\_flags\_decoder.c and function:  extern LZMA\_API(lzma\_ret)  lzma\_stream\_footer\_decode(lzma\_stream\_flags \*options, const uint8\_t \*in)  !!!             // Magic      if (memcmp(in + sizeof(uint32\_t) \* 2 + LZMA\_STREAM\_FLAGS\_SIZE, lzma\_footer\_magic, sizeof(lzma\_footer\_magic)) != 0)          return LZMA\_FORMAT\_ERROR;  // Functions are extremely similar as can be seen (inputs are nearly the same, and processing should also be very similar)  // Again, the check was not activated in ClamAV due to disabling crc checks. | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-126-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // same as 125 even with different error-inducing input  // For 125: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '01', 'ff', 'ff', 'ff']  // For 126: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '10', 'ff', 'ff'] | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-127-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  **For ClamAV:**  // Skipped Check at line 888 in case: XZ\_STATE\_STREAM\_FOOTER of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  à  #ifdef SYMBEX\_DISABLE\_CRCCHECK  #else            if (!Xz\_CheckFooter(p->streamFlags, p->indexSize, p->buf))              return SZ\_ERROR\_CRC;  #endif  vs  **For XZ:**  // Checking to see if the Index Size stored in the Stream Footer matches the real size of the Index field.  !!!  if (lzma\_index\_hash\_size(coder->index\_hash)                  != footer\_flags.backward\_size)              return LZMA\_DATA\_ERROR;  // Root of error happens from 125 and 126, as the following function in file stream\_decoder.c at line 305 generates a faulty &footer\_flags, triggering the error above:  const lzma\_ret ret = lzma\_stream\_footer\_decode(                  &footer\_flags, coder->buffer); | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-130-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // same as 125 even with different error-inducing input  // For 130: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '00', 'ff', 'ff'] | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-61-subfuzz-1-dis-1.xz | Out: 1: | Out: 0 |
| Reason for Discrepancy:  // byte position 25 (\*in\_pos) == 25  **For ClamAV:**  / / Skipped Check at line 752 in case: XZ\_STATE\_BLOCK\_FOOTER of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  // skipped check due to:  #ifdef SYMBEX\_DISABLE\_CRCCHECK            src++;  #else            if (\*src++ != 0)              return SZ\_ERROR\_CRC;  #endif  AND  #ifdef SYMBEX\_DISABLE\_CRCCHECK  #else              if (XzCheck\_Final(&p->check, digest) && memcmp(digest, p->buf, checkSize) != 0)                return SZ\_ERROR\_CRC;  #endif  vs  **For XZ:**  // Check Code padding for correct code structure at line 140 in block\_decoder.c and function:  static lzma\_ret  block\_decode(lzma\_coder \*coder, const lzma\_allocator \*allocator,          const uint8\_t \*restrict in, size\_t \*restrict in\_pos,          size\_t in\_size, uint8\_t \*restrict out,          size\_t \*restrict out\_pos, size\_t out\_size, lzma\_action action)  !!!  case SEQ\_PADDING:          // Compressed Data is padded to a multiple of four bytes.          while (coder->compressed\_size & 3) {              if (\*in\_pos >= in\_size)                  return LZMA\_OK;              // We use compressed\_size here just get the Padding              // right. The actual Compressed Size was stored to              // coder->block already, and won't be modified by              // us anymore.              ++coder->compressed\_size;              if (in[(\*in\_pos)++] != 0x00)                  return LZMA\_DATA\_ERROR;          }  (gdb) print in[25]  $7 = 1 '\001'  (gdb) print in[26]  $8 = 0 '\000'  (gdb) print in[27]  $9 = 0 '\000'  ////////////////////////////////////////////////////  // padding error, not a multiple of four bytes??  // error checking skipped in clamav:  #ifdef SYMBEX\_DISABLE\_CRCCHECK            src++;  #else | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-64-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // Same as 61, but for byte position 26  // byte position (\*in\_pos) == 26  #ifdef SYMBEX\_DISABLE\_CRCCHECK            src++;  #else  if (in[(\*in\_pos)++] != 0x00)         return LZMA\_DATA\_ERROR; | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-67-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy: Reason for Discrepancy:  // Same as 61 and 64, but for byte position 27  // byte position (\*in\_pos) == 27  #ifdef SYMBEX\_DISABLE\_CRCCHECK            src++;  #else  if (in[(\*in\_pos)++] != 0x00)         return LZMA\_DATA\_ERROR;  // Why only positions 25, 26, 27? ->  // while (coder->compressed\_size & 3):  // coder->compressed\_size = 1 = 001 // turn 1 (25)  // & 3 = 011 => 001 = 1  // coder->compressed\_size = 2 = 010 // turn 2 (26)  // & 3 = 011 => 001 = 1  // coder->compressed\_size = 3 = 011 // turn 3 (27)  // & 3 = 011 => 011 = 3  // coder->compressed\_size = 4 = 100 // turn 4 (28)  // & 3 = 011 => 000 = 0  //Seems deliberate to only pass 3 times | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-7-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  **For ClamAV:**  / / Missing Check at line 742 in case: XZ\_STATE\_BLOCK\_HEADER of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  à Continuing into another function  SRes XzBlock\_Parse(CXzBlock \*p, const Byte \*header)  vs  **For XZ:**  // Check Code padding for correct code structure at line 77 in block\_header\_decoder.c and function:  extern LZMA\_API(lzma\_ret)  lzma\_block\_header\_decode(lzma\_block \*block,  const lzma\_allocator \*allocator, const uint8\_t \*in)  !!!  \*in = \*header = “\002\004!\001\020”  Xz has additional feature:  // Check for unsupported flags.      if (in[1] & 0x3C)          return LZMA\_OPTIONS\_ERROR;  (gdb) print header[1]  $4 = 4 '\004'  (gdb) print /t header[1]  $5 = 000100  (gdb) print /t 0x3C  $6 = 111100  in[1] & 0x3C = 00100 = 4 != 0  // possible overlap of data that XZ notices | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-98-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // Nothing special, no immediate returns, ClamAV just has it in a bigger while loop.  // \*in\_pos == 33  **For ClamAV:**  // Skipped Check at lines 798, 827, 837 in case: XZ\_STATE\_STREAM\_INDEX of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  à  #ifdef SYMBEX\_DISABLE\_CRCCHECK            src++;            p->pos++;  #else            if (\*src++ != p->buf[p->pos++])              return SZ\_ERROR\_CRC;  #endif  AND  #ifdef SYMBEX\_DISABLE\_CRCCHECK  #else              if (b != 0)                return SZ\_ERROR\_CRC;  #endif  AND  #ifdef SYMBEX\_DISABLE\_CRCCHECK  #else              if ((p->sha))                  cl\_finish\_hash(p->sha, digest);              if (memcmp(digest, p->shaDigest, SHA256\_DIGEST\_SIZE) != 0)                return SZ\_ERROR\_CRC;  #endif  vs  **For XZ:**  // Check if this is the last byte of a multibyte integer at line 59 of vli\_decoder.c and function:  extern LZMA\_API(lzma\_ret)  lzma\_vli\_decode(lzma\_vli \*restrict vli, size\_t \*vli\_pos,          const uint8\_t \*restrict in, size\_t \*restrict in\_pos,          size\_t in\_size)  !!!          if ((byte & 0x80) == 0) {  …  } | |
| data/xz/20211125\_135535\_xz-5.2.2-clamav-0.99.2\_tgt-0\_all/xz-5.2.2-clamav-0.99.2-tgt-0-ivtpos-99-subfuzz-1-dis-1.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // \*in\_pos == 33  **For ClamAV:**  // Skipped Check at line 798 in case: XZ\_STATE\_STREAM\_INDEX of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  vs  **For XZ:**  // Continuation of 98          if ((byte & 0x80) == 0) {  // We don't allow using variable-length integers as  padding i.e. the encoding must use the most the compact form.              if (byte == 0x00 && \*vli\_pos > 1)                  return LZMA\_DATA\_ERROR;  // \*in\_pos == 33  ClamAV probably has the check disabled due to  #ifdef SYMBEX\_DISABLE\_CRCCHECK | |
| data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000001-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-0-subfuzz-1-dis-3.xz  or  data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000002-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-0-subfuzz-2-dis-3.xz | Out: 1 | Out: 26 |
| Reason for Discrepancy:  **For ClamAV:**  // Extra condition in Check at line 693 in case: XZ\_STATE\_STREAM\_HEADER of function in file XzDec.c:  SRes XzUnpacker\_Code(CXzUnpacker \*p, Byte \*dest, SizeT \*destLen, const Byte \*src, SizeT \*srcLen, int finishMode, ECoderStatus \*status)  !!!  if (p->pos < XZ\_STREAM\_HEADER\_SIZE)          {            if (p->pos < XZ\_SIG\_SIZE && \*src != XZ\_SIG[p->pos]) // XZ\_SIG\_SIZE == 6 ; where does XZ\_SIG come from???              return SZ\_ERROR\_NO\_ARCHIVE;  // Checks to see if the input is of .xz format  // p->pos == 0  vs  **For XZ:**  // Checks to see if the input is of .xz format more quickly, at line 359 of function is\_format\_xz in file coder.c:  /// Return true if the data in in\_buf seems to be in the .xz format.  static bool  is\_format\_xz(void)  {      // Specify the magic as hex to be compatible with EBCDIC systems.      static const uint8\_t magic[6] = { 0xFD, 0x37, 0x7A, 0x58, 0x5A, 0x00 };      return strm.avail\_in >= sizeof(magic)              && memcmp(in\_buf.u8, magic, sizeof(magic)) == 0;  }  // No actual discrepancy even though PATHDIFF “finds” one, potential PATHDIFF bug? | |
| data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000101-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-1-subfuzz-1-dis-3.xz  or  data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000102-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-1-subfuzz-2-dis-3.xz | Out: 1 | Out: 26 |
| Reason for Discrepancy:  // Same as above (id-1000000001), but for ClamAV, p->pos == 1 | |
| data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000201-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-2-subfuzz-1-dis-3.xz  or  data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000202-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-2-subfuzz-2-dis-3.xz | Out: 1 | Out: 26 |
| Reason for Discrepancy:  // Same as above (id-1000000001), but for ClamAV, p->pos == 2 | |
| data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000301-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-3-subfuzz-1-dis-3.xz  or  data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000302-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-3-subfuzz-2-dis-3.xz | Out: 1 | Out: 0 |
| Reason for Discrepancy:  // Same as above (id-1000000001), but for ClamAV, p->pos == 3 | |
| data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000401-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-4-subfuzz-1-dis-3.xz  or  data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000402-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-4-subfuzz-2-dis-3.xz | Out: 1 | Out: 26 |
| Reason for Discrepancy:  // Same as above (id-1000000001), but for ClamAV, p->pos == 4 | |
| data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000501-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-5-subfuzz-1-dis-3.xz  or  data/xz/20220510\_115422\_xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e\_tgt-1\_all/testcases/id-1000000502-xz-5.2.2-clamav-0.99.2-update-1-rank-1-e240e-tgt-1-ivtpos-5-subfuzz-2-dis-3.xz | Out: 1 | Out: 26 |
| Reason for Discrepancy:  // Same as above (id-1000000001), but for ClamAV, p->pos == 5 | |
| data/xz/20211125\_135555\_xz-5.2.2-clamav-0.99.2\_tgt-1\_all/xz-5.2.2-clamav-0.99.2-tgt-1-ivtpos-49-subfuzz-1-dis-1.xz | Out: 0 | Out: 26 |
| Reason for Discrepancy:  New Property Dictionay size (>200000000) allocation exceeds CLI\_MAX\_ALLOCATION ( 182\*1024\*1024 = 190840832 bytes) for clamav. | |

**Additional Info: IVTPOS 7**

Input Bytes for IVTPOS 7: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '04', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '01', '00', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

|  |  |
| --- | --- |
| ClamAV | XZ |
| A screen shot of a computer program  Description automatically generated | A computer screen shot of a program  Description automatically generated   * Missing Check for unsupported flags in ClamAV * In both programs, header[pos++] == in[1] == 0x04 |

**Additional Info: IVTPOS 61/64/67 \*\***

Input Bytes for IVTPOS 61: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '01', '00', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

Input Bytes for IVTPOS 64: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '01', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

Input Bytes for IVTPOS 67: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '01', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

|  |  |
| --- | --- |
| ClamAV | XZ |
| A screen shot of a computer program  Description automatically generated | A screenshot of a computer program  Description automatically generated |
| **Line 754:**  (gdb) print p->packSize  $1 = 1  (gdb) print p->alignPos  $2 = 0  // (p->packSize + p->alignPos) == 1  **Line 756:**  (gdb) print \*srcLen  $3 = 25  (gdb) print \*srcLen  $4 = 26  // Update the current byte position (however, src is still a pointer to byte 25, haven’t updated yet)  **Line 757:**  (gdb) print p->alignPos  $6 = 1  // Update alignPos and thus (p->packSize + p->alignPos) == 2  **Line 761 (is skipped):**  (gdb) print \*src  $7 = 1 '\001'  // Not a valid byte, should be 0x00  // Should have detected an error and would have updated \*src to point to the next byte position (26), but due to disabling CRC Checks, it isn’t able to catch the error.  // Instead goes to Line 759 | **Line 130:**  (gdb) print coder->compressed\_size  $1 = 1  **Line 131:**  // Simple check to see if current position exceeded total input size of 52 bytes, not especially relevant  **Line 134:**  (gdb) print \*in\_pos  $2 = 25  **Line 138:**  (gdb) print coder->compressed\_size  $3 = 2  // compressed\_size == (p->packSize + p->alignPos) == 2  **Line 140:**  (gdb) print in[\*in\_pos] == \*src  $4 = 1 '\001'  // Not a valid byte, should be 0x00  // Detects it and updates \*in\_pos to the next byte position (26)  // Returns LZMA\_DATA\_ERROR |

// Why only positions 25, 26, 27? ->

// while (coder->compressed\_size & 3):

// coder->compressed\_size = 1 = 001 // turn 1 (25)

// & 3 = 011 => 001 = 1

// coder->compressed\_size = 2 = 010 // turn 2 (26)

// & 3 = 011 => 001 = 1

// coder->compressed\_size = 3 = 011 // turn 3 (27)

// & 3 = 011 => 011 = 3

// coder->compressed\_size = 4 = 100 // turn 4 (28)

// & 3 = 011 => 000 = 0

//Seems deliberate to only pass 3 times

**Additional Info: IVTPOS 98/99/100\*\***

Input Bytes for IVTPOS 98: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '80', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

Input Bytes for IVTPOS 99: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '00', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

Input Bytes for IVTPOS 100: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '03', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

For 98/99/100, byte 33 is supposed to be ‘01’

|  |  |
| --- | --- |
| ClamAV | XZ |
| Changes the buffer, p->numBlocks, to the correct starting value of 1 on line 718  Old value = "\000\201\000\000\020\000\000\000\250p\216\206", '\000' <repeats 1011 times>  New value = "\000\001\000\000\020\000\000\000\250p\216\206", '\000' <repeats 1011 times>  Xz\_WriteVarInt (buf=0x5618e5dc8a69 "\001", v=0) at ../../libclamav/7z/Xz.c:27  A computer screen with white text  Description automatically generated  A screen shot of a computer program  Description automatically generated | Part of function lzma\_vli\_decode (&index\_hash->remaining = \*vli)  A computer screen shot of a program code  Description automatically generated  Index\_hash->blocks.count is equivalent to p->numBlocks = 1 (it’s the buffer to confirm that there is no data corruption)  A screen shot of a computer code  Description automatically generated |

**Additional Info: IVTPOS 104/105/106/110/111**

Input Bytes for IVTPOS 104: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '80', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

Input Bytes for IVTPOS 105: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '00', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

Input Bytes for IVTPOS 106: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '01', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

For 104/105/106, byte 34 is supposed to be ‘11’

Input Bytes for IVTPOS 110: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '11', '80', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

Input Bytes for IVTPOS 111: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '11', '01', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

For 110/111, byte 35 is supposed to be ‘00’

|  |  |
| --- | --- |
| ClamAV | XZ |
| 816 in ../../libclamav/7z/XzDec.c  (gdb) print \*srcLen  $9 = 34  (gdb) n  817 in ../../libclamav/7z/XzDec.c  (gdb) print \*srcLen  $10 = 36  (gdb)  Completely skips checking over byte 35, and also does not check for redundancy in byte 34  A computer screen shot of a program  Description automatically generated  No similarities | Part of function lzma\_vli\_decode (&index\_hash->remaining = \*vli)  A computer screen shot of a program code  Description automatically generated  Index\_hash->blocks.count is equivalent to p->numBlocks = 1 (it’s the buffer to confirm that there is no data corruption)  A computer screen shot of a program code  Description automatically generated |

**Additional Info: IVTPOS 114/118\*\***

Input Bytes for IVTPOS 104: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '15', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

Input Bytes for IVTPOS 105: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '05', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '01', '59', '5a']

|  |  |
| --- | --- |
| ClamAV | XZ |
| index\_hash->records.blocks\_size == Byte digest  index\_hash->blocks.blocks\_size == p->shaDigest (buffer to confirm no data corruption)  (gdb) print memcmp(digest, p->shaDigest, SHA256\_DIGEST\_SIZE)  No symbol "SHA256\_DIGEST\_SIZE" in current context.  (gdb) print memcmp(digest, p->shaDigest, 32)  $11 = 17  A computer screen shot of a program code  Description automatically generated | A computer screen shot of a program  Description automatically generated  A computer screen shot of a program code  Description automatically generated |

**Additional Info: IVTPOS 125/126/127\*\***

Input Bytes for IVTPOS 125: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '01', 'ff', 'ff', 'ff']

Input Bytes for IVTPOS 126: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '10', 'ff', 'ff']

Input Bytes for IVTPOS 127: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '00', '00', '00', '00', '00', '00', 'ff', 'ff']

Bytes 48, 49, 50, 51 supposed to be respectively ‘00’, ‘01’, ‘59’, ‘5a’

|  |  |
| --- | --- |
| ClamAV | XZ |
| A screen shot of a computer program  Description automatically generated  A computer screen shot of a program  Description automatically generated  // Xz\_CheckFooter disabled, causing a discrepancy | A computer screen with text on it  Description automatically generated  A computer screen shot of a program  Description automatically generated  sizeof(uint32\_t) == 4, if you replace this you will see the same numbers as in ClamAV |

**Additional Info: IVTPOS 130**

Input Bytes for IVTPOS 130: ['fd', '37', '7a', '58', '5a', '00', '00', '01', '69', '22', 'de', '36', '02', '00', '21', '01', '10', '00', '00', '00', 'a8', '70', '8e', '86', '00', '00', '00', '00', '00', '00', '00', '00', '00', '01', '11', '00', '3b', '96', '5f', '73', '90', '42', '99', '0d', '01', '00', '00', '00', '00', '00', 'ff', 'ff']

Bytes 44, 49, 50, 51 supposed to be respectively ‘00’, ‘01’, ‘59’, ‘5a’

|  |  |
| --- | --- |
| ClamAV | XZ |
| A computer screen shot of a program  Description automatically generated  A computer screen shot of a program  Description automatically generated | A screen shot of a computer  Description automatically generated |