Base Language - Bug #6399

ENCRYPT/DECRYPT built-in function results different than 4GL when IV is shorter than required

05/24/2022 09:55 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Theodoros Theodorou	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 05/24/2022 09:55 AM - Greg Shah

From #5915-104:

> FWD implementation for ENCRYPT produces different result than 4GL when the cipher requires IV and the provided IV is shorter than required. This includes AES in CBC/CFB/OFB modes and IV shorter than 16, DES3/DES in CBC/CFB/OFB modes and IV is shorter than 8. It is unclear what IV 4GL provides to the cipher in these cases. Maybe it is something like described in #5915-40.

#3 - 01/13/2023 02:18 PM - Greg Shah

- Assignee set to Theodoros Theodorou

#4 - 01/16/2023 11:51 AM - Greg Shah

The runtime implementation of this is in SecurityOps.encrypt()/decrypt(). There should be no conversion changes needed (the conversion occurs in rules/convert/builtin_functions.rules, search on prog.kw_encrypt or prog.kw_decrypt to find where we define the conversion mapping.

#5 - 02/09/2023 09:54 AM - Theodoros Theodorou

- Status changed from New to WIP

#6 - 02/13/2023 01:45 PM - Theodoros Theodorou

I have read some existing tests under testcases/security/security_policy and I tried to understand them. I tried many things but I couldn't recreate the issue. I used AES_CFB_128 and the IV was set to 2 (shorter than 16) on the following example. I executed the code below and it gave me identical results with FWD and OpenEdge.

```
def var k      as raw       no-undo.
def var iv     as raw       no-undo.
def var s      as character no-undo.
def var encr   as memptr    no-undo.
def var decr   as memptr    no-undo.
def var enchar as longchar  no-undo.
def var dechar as longchar  no-undo.

s = 'Hello world!'.
k = generate-pbe-key('k').
put-byte(iv, 1) = 2.
```

```
message 'Key: '               + string(k).
message 'IV length: '         + string(length(iv)).
message 'IV: '                + string(iv).
message 'IV string length: ' + string(length(string(iv))).

encr = encrypt(s, k, iv, 'AES_CFB_128').
decr = decrypt(encr, k, iv, 'AES_CFB_128').

enchar = base64-encode(encr).
message 'Encoded string: ' + string(enchar).

copy-lob from decr to dechar.
message 'Decoded string: ' + string(dechar).
```

I would like to ask for some help recreating the issue (if the issue still exists).

#7 - 02/13/2023 02:21 PM - Constantin Asofiei

From #5915-40:

> The problem is with generate-pbe-key, where AES_CBC_256 is used for this key (32 length). There is this interesting note here: https://community.progress.com/s/question/0D54Q00007ztLiuSAE/generatepbekey-inconsistent-between-net-for-large-keys
>
> ```
> Just in case anyone comes across this and needs an answer. OpenEdge uses the OpenSSL encryption library un
> der the hood. Specifically, the GENERATE-PBE-KEY() method returns the same output as OpenSSL's EVP_BytesTo
> Key method. There's a C# version available here on Github (not my code).
> ```
>
> With the OpenSSL notes here: https://www.openssl.org/docs/manmaster/man3/EVP_BytesToKey.html
>
> ```
> KEY DERIVATION ALGORITHM
> The key and IV is derived by concatenating D_1, D_2, etc until enough data is available for the key and IV
> . D_i is defined as:
>
>        D_i = HASH^count(D_(i-1) || data || salt)
> where || denotes concatenation, D_0 is empty, HASH is the digest algorithm in use, HASH^1(data) is simply
> HASH(data), HASH^2(data) is HASH(HASH(data)) and so on.
>
> The initial bytes are used for the key and the subsequent bytes for the IV.
> ```
>
> This may explain this comment:
>
> ```
>      // Generate tail bytes, which is not  PBKDF1 compliant but 4GL does it in some way.
>      // It is unclear how exactly it is implemented so we use the first 8 bytes of the
>      // previous step as  a salt for a next chunk.
> ```

I think the issue in this task was fixed in trunk rev 13659. A customer's scenario which triggered this solution was:

```
def var lcdata as longchar.
def var lc as longchar.
def var rkey as raw.
def var ckey as char.
def var mMemData as memptr.
def var mMemPtr as memptr.

lcdata = "<encrypted-data>".
ckey = "<key>".

security-policy:symmetric-encryption-algorithm = "AES_CBC_256":u.
security-policy:symmetric-encryption-iv = ?.
rKey = generate-pbe-key(cKey).

// set the encryption key
security-policy:symmetric-encryption-key = rKey.
mMemData      = base64-decode(lcData).
set-size(mMemPtr) = 0.
mMemPtr = decrypt(mMemData, rKey).
if get-size(mMemPtr) = 0
   then lc = "n/a".
   else copy-lob from mMemPtr to lc.

/* clear memory space after use */
set-size(mMemPtr) = 0.
set-size(mMemData) = 0.

message string(lc).
```

I don't recall exactly how <encrypted-data> and <key> was computed by the customer's code.

#8 - 02/13/2023 02:26 PM - Igor Skornyakov

As far as I rememeber after the Constantin's fix we are 100% binary compatible with 4GL with respect to the encryption key. and the problem is with

short IV only.
May be the analysys of the OpenSSL source code can give a clue.

#9 - 02/14/2023 08:38 AM - Theodoros Theodorou

I tried many combinations with AES_CBC_256. Some of them can be found below. I keep getting identical results with FWD and OpenEdge.

This example is copied and modified from #5915-39:

```
def var lcdata as longchar.
def var lc as longchar.
def var rkey as raw.
def var iv as raw.
def var ckey as char.
def var mMemData as memptr.
def var mMemPtr as memptr.

lcdata = "ZQSXDX+ZT4vlMRmgQr4JzcG9+Lw==".
ckey = "852564e3-27eb-3a99-3814-bb1830a2aa63".
put-string(iv, 1) = '6'.

security-policy:symmetric-encryption-algorithm = "AES_CBC_256":u.
security-policy:symmetric-encryption-iv = iv.
rKey = generate-pbe-key(cKey).

message string(lcdata).
message ckey.
message string(rkey).
message string(iv).

// set the encryption key
security-policy:symmetric-encryption-key = rKey.
lcData        = substring(lcData, 6).  // skip ZQSXD chars
mMemData      = base64-decode(lcData).
set-size(mMemPtr) = 0.
mMemPtr = decrypt(mMemData, rKey).
if get-size(mMemPtr) = 0
   then lc = "n/a".
   else copy-lob from mMemPtr to lc.

/* clear memory space after use */
set-size(mMemPtr) = 0.
set-size(mMemData) = 0.

message string(lcData).
message string(lc).
```

Another example:

```
def var k      as raw       no-undo.
def var iv     as raw       no-undo.
def var s      as character no-undo.
def var encr   as memptr    no-undo.
def var decr   as memptr    no-undo.
def var enchar as longchar  no-undo.
def var dechar as longchar  no-undo.

security-policy:symmetric-encryption-algorithm = 'AES_CBC_256'.
s = 'Hello world!'.
k = generate-pbe-key('1').
put-string(iv, 1) = '2'.

message 'Key: '              + string(k).
message 'IV length: '        + string(length(iv)).
message 'IV: '               + string(iv).
message 'IV string length: ' + string(length(string(iv))).

security-policy:symmetric-encryption-iv = iv.
encr = encrypt(s, k).
```

```
decr = decrypt(encr, k).

enchar = base64-encode(encr).
message 'Encoded string: ' + string(enchar).

copy-lob from decr to dechar.
message 'Decoded string: ' + string(dechar).
```

Constantin Asofiei wrote:

> I think the issue in this task was fixed in trunk rev 13659.

I think there is a big chance that the bug has already been fixed.

#10 - 02/14/2023 08:57 AM - Greg Shah

I'm OK with closing this task. Before we do that, please do/consider the following:

- Gather all of your various testcases and make sure the different variants are all represented the set of programs.
- Make changes to ensure that the processing is 100% automated. There should be no interactive usage.
- Let's determine how to locate/integrate these in the testcases project.
- What do we need to do to ensure that all of the encryption/decryption tests can be run as a single set?

#11 - 02/14/2023 09:59 AM - Igor Skornyakov

- File crypto.p added

I've used the attached program for testing cryptography.
May be it can be useful.

#12 - 02/16/2023 04:56 AM - Theodoros Theodorou

Thank you Igor for your help regarding the test. I modified it a bit and added it under testcases/security/security_policy/attributes/test-encryption-iv.p. I forgot to mention that my p2j project is on the branch 6421a, which has a fix for #6421. The test has identical results with FWD and OE.

#13 - 02/16/2023 07:15 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from WIP to Closed

Files

crypto.p	4.98 KB	02/14/2023	Igor Skornyakov