Lab8 Report of PoRE

Student ID 18307130012

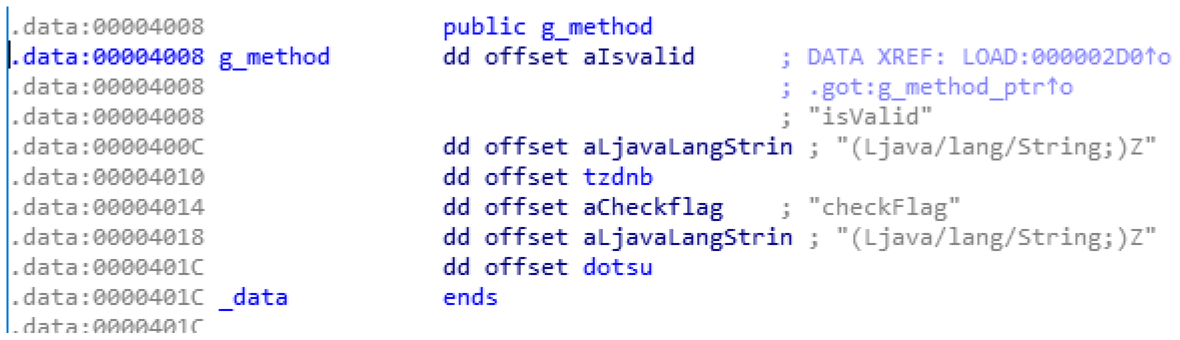
Student Name 管箫 Guan Xiao

- **Your answer(a simple string)**

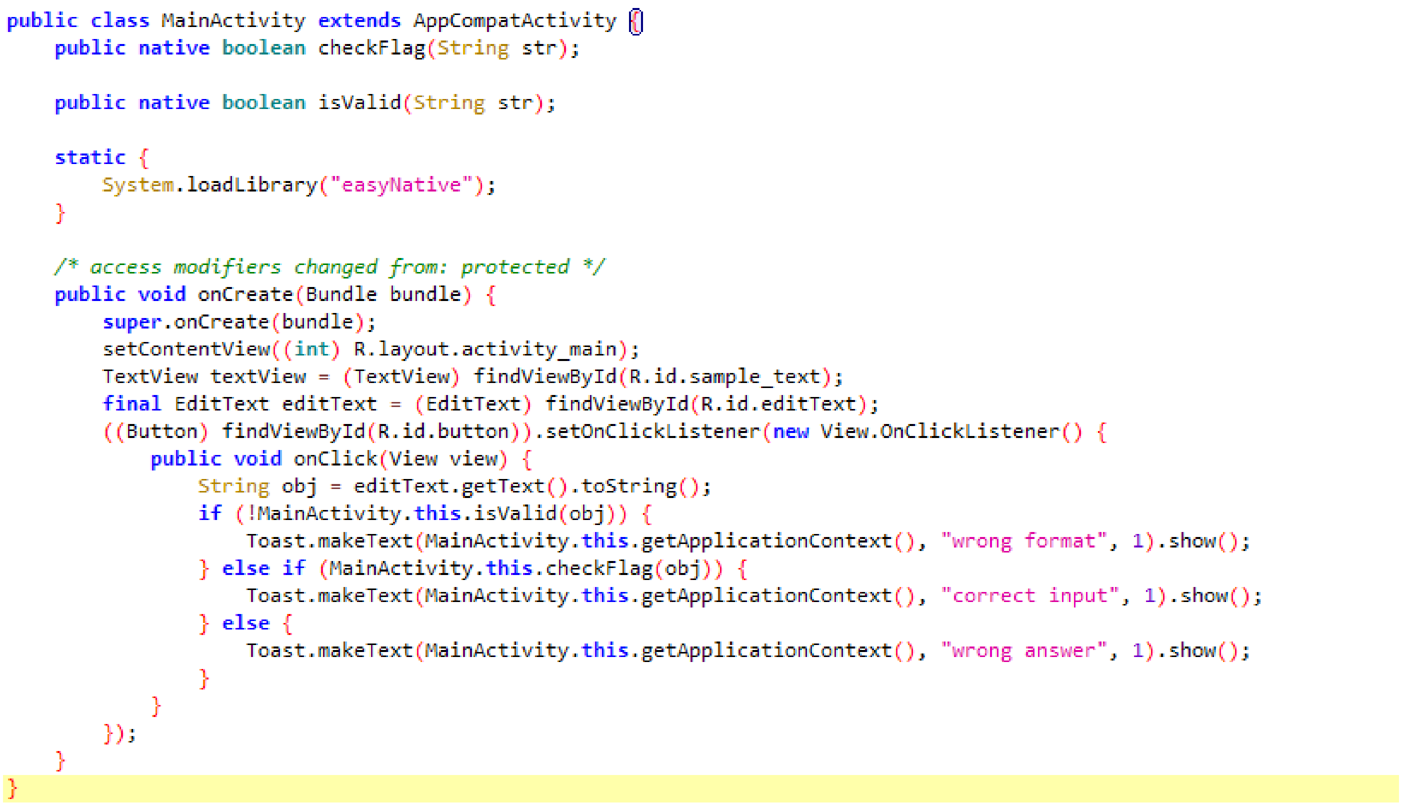
Na7Iv3_b@sE64_r0T13_c0de

- **Describe how you reversed the native code and what the code do**
- **Describe how you get the right answer**

① Find the native function name: checkFlag(String str) & isValid(String str)



```
.data:00004008                 public g_method
.data:00004008 g_method        dd offset aIsvalid      ; DATA XREF: LOAD:000002D0↑o
.data:00004008                                         ; .got:g_method_ptr↑o
.data:00004008                                         ; "isValid"
.data:0000400C                 dd offset aLjavaLangStrin ; "(Ljava/lang/String;)Z"
.data:00004010                 dd offset tzdnb
.data:00004014                 dd offset aCheckflag    ; "checkFlag"
.data:00004018                 dd offset aLjavaLangStrin ; "(Ljava/lang/String;)Z"
.data:0000401C                 dd offset dotsu
.data:0000401C _data           ends
.data:0000401C
```



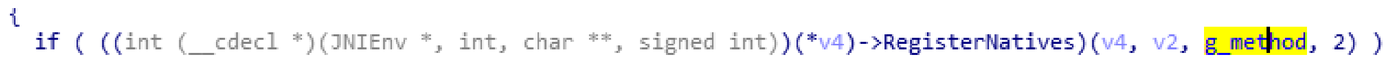
```java
public class MainActivity extends AppCompatActivity {
    public native boolean checkFlag(String str);

    public native boolean isValid(String str);

    static {
        System.loadLibrary("easyNative");
    }

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_main);
        TextView textView = (TextView) findViewById(R.id.sample_text);
        final EditText editText = (EditText) findViewById(R.id.editText);
        ((Button) findViewById(R.id.button)).setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                String obj = editText.getText().toString();
                if (!MainActivity.this.isValid(obj)) {
                    Toast.makeText(MainActivity.this.getApplicationContext(), "wrong format", 1).show();
                } else if (MainActivity.this.checkFlag(obj)) {
                    Toast.makeText(MainActivity.this.getApplicationContext(), "correct input", 1).show();
                } else {
                    Toast.makeText(MainActivity.this.getApplicationContext(), "wrong answer", 1).show();
                }
            }
        });
    }
}
```

②Use IDA to analysis the onLoad function, find it rename the isValid and checkFlag function

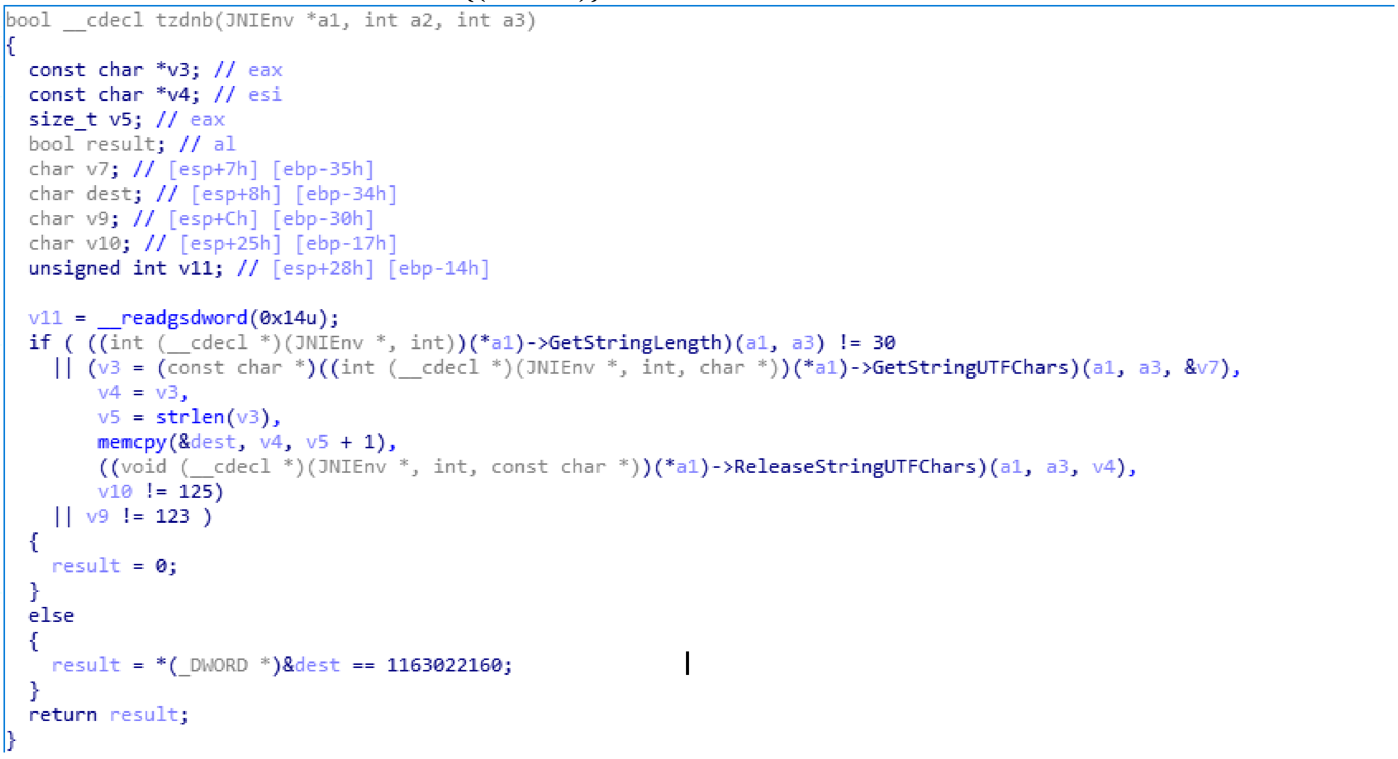


```
{
  if ( ((int (__cdecl *)(JNIEnv *, int, char **, signed int))(*v4)->RegisterNatives)(v4, v2, g_method, 2) )
```

③Check the .data table, find the real function is “tzdnb” and “dotsu”。

④Check the tzdnb function, find it check the input string’s format.

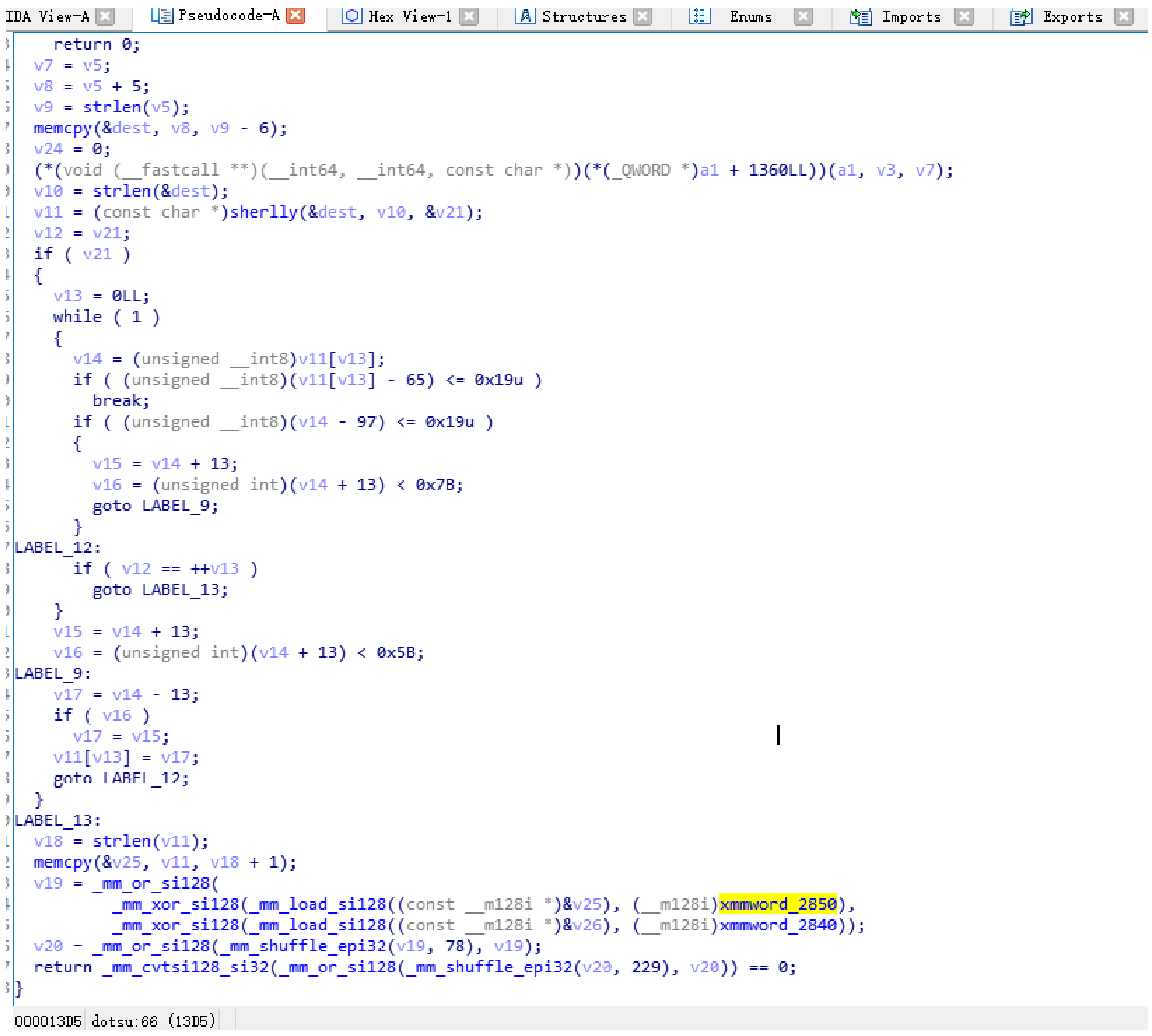
The format should be “PORE{(24char)}”



```
bool __cdecl tzdnb(JNIEnv *a1, int a2, int a3)
{
  const char *v3; // eax
  const char *v4; // esi
  size_t v5; // eax
  bool result; // al
  char v7; // [esp+7h] [ebp-35h]
  char dest; // [esp+8h] [ebp-34h]
  char v9; // [esp+Ch] [ebp-30h]
  char v10; // [esp+25h] [ebp-17h]
  unsigned int v11; // [esp+28h] [ebp-14h]

  v11 = __readgsdword(0x14u);
  if ( ((int (__cdecl *)(JNIEnv *, int))(*a1)->GetStringLength)(a1, a3) != 30
    || (v3 = (const char *)((int (__cdecl *)(JNIEnv *, int, char *))(*a1)->GetStringUTFChars)(a1, a3, &v7),
        v4 = v3,
        v5 = strlen(v3),
        memcpy(&dest, v4, v5 + 1),
        ((void (__cdecl *)(JNIEnv *, int, const char *))(*a1)->ReleaseStringUTFChars)(a1, a3, v4),
        v10 != 125)
    || v9 != 123 )
  {
    result = 0;
  }
  else
  {
    result = *(_DWORD *)&dest == 1163022160;
  }
  return result;
}
```

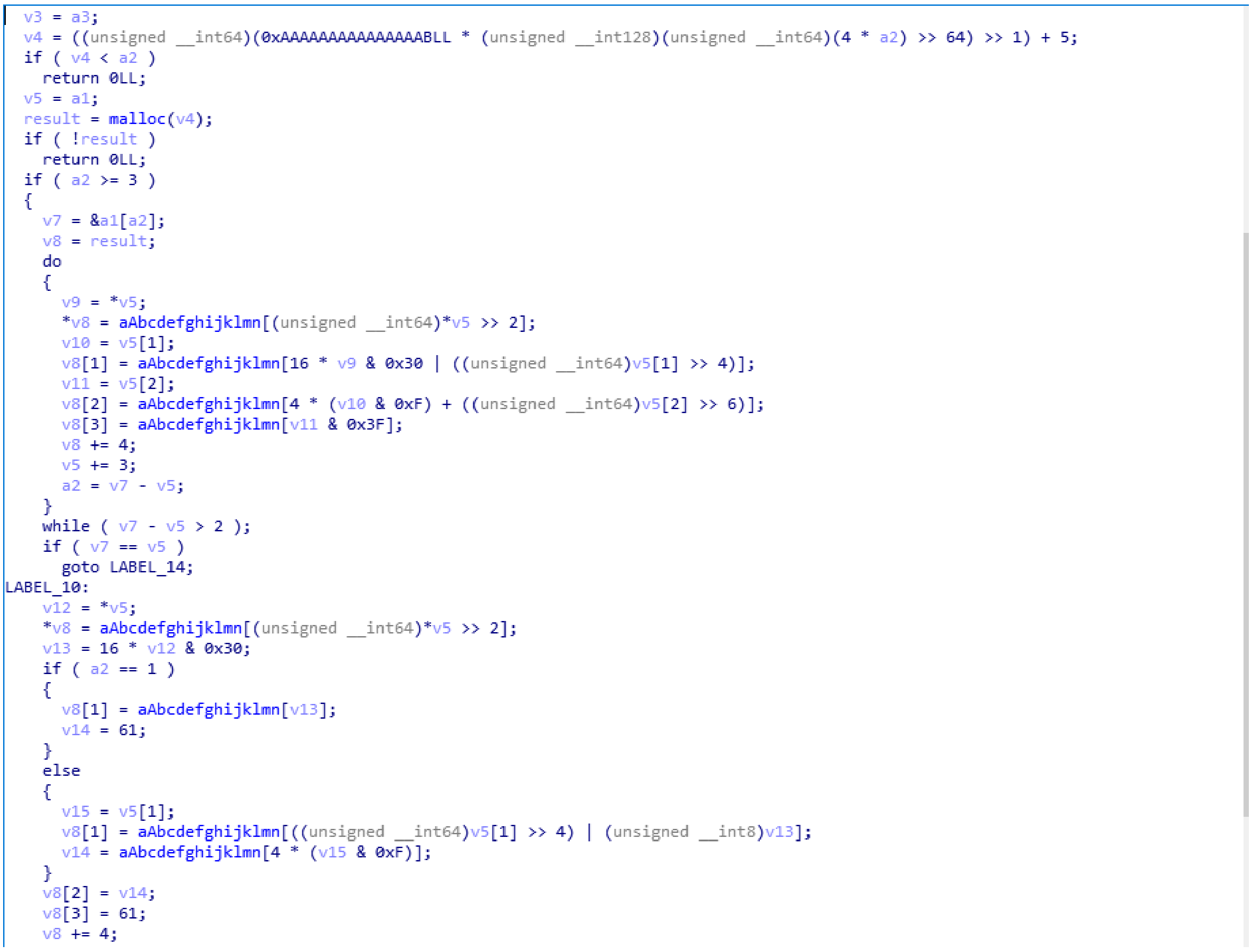
⑤Research the dotsu function



```
    return 0;
  v7 = v5;
  v8 = v5 + 5;
  v9 = strlen(v5);
  memcpy(&dest, v8, v9 - 6);
  v24 = 0;
  (*(void (__fastcall **)(__int64, __int64, const char *))(*(_QWORD *)a1 + 1360LL))(a1, v3, v7);
  v10 = strlen(&dest);
  v11 = (const char *)sherlly(&dest, v10, &v21);
  v12 = v21;
  if ( v21 )
  {
    v13 = 0LL;
    while ( 1 )
    {
      v14 = (unsigned __int8)v11[v13];
      if ( (unsigned __int8)(v11[v13] - 65) <= 0x19u )
        break;
      if ( (unsigned __int8)(v14 - 97) <= 0x19u )
      {
        v15 = v14 + 13;
        v16 = (unsigned int)(v14 + 13) < 0x7B;
        goto LABEL_9;
      }
LABEL_12:
      if ( v12 == ++v13 )
        goto LABEL_13;
    }
    v15 = v14 + 13;
    v16 = (unsigned int)(v14 + 13) < 0x5B;
LABEL_9:
    v17 = v14 - 13;
    if ( v16 )
      v17 = v15;
    v11[v13] = v17;
    goto LABEL_12;
  }
LABEL_13:
  v18 = strlen(v11);
  memcpy(&v25, v11, v18 + 1);
  v19 = _mm_or_si128(
          _mm_xor_si128(_mm_load_si128((const __m128i *)&v25), (__m128i)xmmword_2850),
          _mm_xor_si128(_mm_load_si128((const __m128i *)&v26), (__m128i)xmmword_2840));
  v20 = _mm_or_si128(_mm_shuffle_epi32(v19, 78), v19);
  return _mm_cvtsi128_si32(_mm_or_si128(_mm_shuffle_epi32(v20, 229), v20)) == 0;
}
```

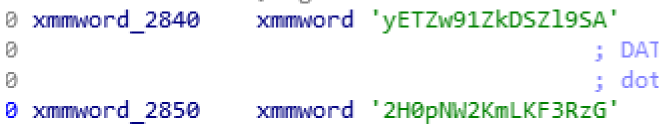
000013D5 dotsu:66 (13D5)

Find its job is to convert all the char in string,
First called sherlly function, then transfer all of them.
From A-M to N-Z, a-m to n-z.



```
  v3 = a3;
  v4 = ((unsigned __int64)(0xAAAAAAAAAAAAAAABLL * (unsigned __int128)(unsigned __int64)(4 * a2) >> 64) >> 1) + 5;
  if ( v4 < a2 )
    return 0LL;
  v5 = a1;
  result = malloc(v4);
  if ( !result )
    return 0LL;
  if ( a2 >= 3 )
  {
    v7 = &a1[a2];
    v8 = result;
    do
    {
      v9 = *v5;
      *v8 = aAbcdefghijklmn[(unsigned __int64)*v5 >> 2];
      v10 = v5[1];
      v8[1] = aAbcdefghijklmn[16 * v9 & 0x30 | ((unsigned __int64)v5[1] >> 4)];
      v11 = v5[2];
      v8[2] = aAbcdefghijklmn[4 * (v10 & 0xF) + ((unsigned __int64)v5[2] >> 6)];
      v8[3] = aAbcdefghijklmn[v11 & 0x3F];
      v8 += 4;
      v5 += 3;
      a2 = v7 - v5;
    }
    while ( v7 - v5 > 2 );
    if ( v7 == v5 )
      goto LABEL_14;
LABEL_10:
    v12 = *v5;
    *v8 = aAbcdefghijklmn[(unsigned __int64)*v5 >> 2];
    v13 = 16 * v12 & 0x30;
    if ( a2 == 1 )
    {
      v8[1] = aAbcdefghijklmn[v13];
      v14 = 61;
    }
    else
    {
      v15 = v5[1];
      v8[1] = aAbcdefghijklmn[((unsigned __int64)v5[1] >> 4) | (unsigned __int8)v13];
      v14 = aAbcdefghijklmn[4 * (v15 & 0xF)];
    }
    v8[2] = v14;
    v8[3] = 61;
    v8 += 4;
```

And we can easily find out the sherlly function is to do Base64 transformation.



```
0 xmmword_2840     xmmword 'yETZw91ZkDSZl9SA'
0                                          ; DAT
0                                          ; dot
0 xmmword_2850     xmmword '2H0pNW2KmLKF3RzG'
```

The final string will be test whether equal to this string.

So, with help of Base64 format calculator, we can easily find out the origin String.