1.3.2 "%%" causes buffer-overflow

- **Bug:** In the format-string, a pair of "%" characters (i.e., "%%") causes a buffer-overflow

In a format-string, "%%" is used to put a "%" in the output-string. "%%" does not correspond to any format-specifier arguments. Both `S...PrintfHost()` and `printf()` work like this.

Each "%%" specified causes an 8-byte buffer-overflow. The maximum buffer-overflow is 64 bytes. The buffer-overflow can be benign, or it can cause the SystemView app to crash.

Problem analysis:

`_VPrintHost()` parses the format-string to find "%" characters. For each "%" found, `va_arg()` is called to get a format-specifier (an argument). Four bytes are returned for each `va_arg()` call, and the four bytes are put in a buffer (`aParas[]`).

The problem is that `_VPrintHost()` interprets "%%" as specifying two format-specifiers (two arguments). `va_arg()` gets called twice for "%%", but `va_arg()` should not be called at all for them.

For each "%%" pair, `va_arg()` will be called for two arguments that do not exist. `va_arg()` does not specify how it works when called for a non-existent argument. Presumably it results in an out-of-bounds access in the stack. (More info is at this link:)
https://wiki.sei.cmu.edu/confluence/display/c/EXP47-C.+Do+not+call+va_arg+with+an+argument+of+the+incorrect+type

For each "%%" pair, the buffer `aParas[]` gets overflowed by 8 bytes. `aParas[]` is a local variable, so it's on the stack.

On most systems, the largest buffer-overflow possible is 64 bytes. By default, `_VPrintHost()` cannot make more than 16 calls to `va_arg()`. (The limit is specified by `SEGGER_SYSVIEW_MAX_ARGUMENTS`.). So, a format-string with eight "%%" pairs will result in a 64-byte overflow (16 * 4 bytes).

An overflow of 64 bytes will likely overwrite something important on the stack. In the code below, the format-string has 8 eight "%%" pairs. So it creates an overflow of 64 bytes. In running the code, the SystemView app crashed on the Windows host.

```
  char ca = 'a';
  SEGGER_SYSVIEW_PrintfHost("%%%%%%%%%%%%%%%%12%c345678", ca);
```

From testing, specifying "%%" in a format string doesn't always cause failure. For example, this call seemed to work OK:

```
  SEGGER_SYSVIEW_PrintfHost("%%12c345678");
```

1.3.3 "%c" with 0x00 terminates the displayed output-string

- **Bug**: using "%c" with 0x00 terminates the displayed output-string

There is a bug in the SystemView app, in how it displays the output-string from a call to `S...PrintfHost()`. The bug occurs when an output-string has a binary-zero prior to the string's final null-terminator. The first binary-zero is incorrectly interpreted as the output-string's final null-terminator, and the rest of the string is not displayed.

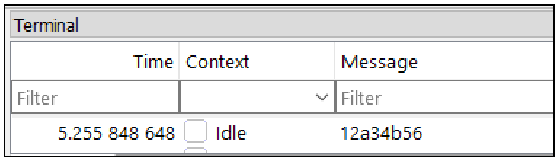
For `S...PrintfHost()`,"%c" is used to specify a single byte for the output-string. The following code uses "%c" three times:

```
    char ca = 'a'; char cb = 'b'; char c0 = 0;
    SEGGER_SYSVIEW_PrintfHost("12%c34%c56%c78", ca, cb, c0);
```

The output-string should be: `12a34b56\078\0`. "\0" is binary zero (0x00). The first "\0" is from the third "%c", and the second "\0" is the output-string's final null-terminator.

When the output-string is displayed in the SystemView app, only the characters prior to the first "\0" are displayed. Output for the above code is shown below. The first binary-zero is incorrectly interpreted as the output-string's final null-terminator, and the rest of the string is not displayed.

The Terminal window:


Terminal

Time	Context	Message
Filter		Filter
5.255 848 648	Idle	12a34b56